明細書

記録媒体、再生装置、記録方法、プログラム、再生方法

技術分野

本発明は、動画ストリーム及びグラフィクスストリームが多重化されたデジタルストリームが記録されている記録媒体と、そのデジタルストリームを再生する再生装置とに関し、特に動画ストリーム及びグラフィクスストリームを再生装置が別個にデコードし、そのデコードにより得られた映像ーグラフィクスを合成することで、再生映像を得る技術に関する。

背景技術

上述した合成技術は、再生装置における言語設定やディスプレィ設定に応じて、グラフィクスを選択して表示することもできる。また必要に応じてグラフィクスを表示したり、消したりという選択も可能である。これらの選択が可能なので、かかる技術は、DVD-Video規格やETSI EN 300 743標準規格のプレーヤモデルに採用されている。上記規格技術においてグラフィクスストリームは、PESパケットの配列であり、PESパケットには、制御情報を格納したもの、グラフィクス本体たるグラフィクスデータを格納したものの2種類がある。制御情報は、グラフィクスデータより前に配置されており、制御情報及びグラフィクスデータの一対で一個のグラフィクス表示を実現する。具体的にいうと、再生装置は制御情報及びグラフィクスデータが順次読み込み、制御情報を解読すると共に、グラフィクスデータをデコードし、デコードにより得られた非圧縮グラフィクスを、制御情報の解読結果に従い、所望の表示タイミングで表示する。

以上のようなグラフィクス表示にあたっては、グラフィクス表示の度に、グラフィクスデータのデコードを行う必要があるので、1つのグラフィクス表示から、次のグラフィクス表示までの間隔は、グラフィクスデータのデコード期間に依存したものとなる。ここでグラフィクスの解像度が高く、デコード期間が長ければ長いほど、グラフィクスの更新間隔も長くなってしまう。

映画における字幕のように、グラフィクスが2～3秒という時間間隔

で更新されるのであれば、多少、グラフィクスの解像度が高く、デコード期間が長くなったとしても問題はない。しかし、アミューズメントの用途にグラフィクスを使用しようとする場合、グラフィクスの表示間隔をもっと短くしたいという要望がでてくる。

ここでアミューズメントの用途とは、登場人物の台詞を示すグラフィクスに、独特の動きを与えて、視聴者の注意をひいたり、色彩を派手に変化させることで、視聴者の意表を突くというものである。かかるグラフィクス表示の実現は、バラエティ色が濃い映像コンテンツの制作現場からの要望が特に強い。

しかしグラフィクスの滑らかな動きを実現するには、グラフィクスをデコードし、表示するという処理を、ディスプレィの一表示期間(NTSC方式の場合は1/29.97sec)という僅かな期間が経過する毎に、行う必要がある。ディスプレィの一表示期間が経過する度に、グラフィクスのデコード一表示を繰り返すというのは、多大な負荷であり、低コストでの普及を想定しているような再生装置のハードウェアスペックでは実現が困難である。

無論、上述したようなグラフィクス動きや色彩変化は、動画像の一コマ一コマにグラフィクスを予め合成しておき、各コマの絵柄の一部としてグラフィクスを取り扱うことで実現可能である。しかし一コマ一コマに、グラフィクスを予め合成しておくといく方法は、再生装置における言語設定やディスプレィ設定に応じて、グラフィクスを選択的に表示するという、融通性を欠く。そのため、かかる方法を採用している限り、将来への展望は見えてこない。

発明の開示

本発明の目的は、グラフィクスを、動画のような滑らかさで動かすことができる記録媒体を提供することである。

上記目的は、動画ストリームとグラフィクスストリームとを多重化することにより得られたデジタルストリームが記録されており、グラフィクスストリームは、パケットの配列であり、パケットには、グラフィクスデータを格納したものと、制御情報を格納したものとがあり、

制御情報は、パケット列において自身より前方に存在するグラフィクスデータを、所定のタイミングで動画ストリームと合成して表示する旨を示す、ことを特徴とする記録媒体により達成される。

制御情報は、自身より前方に存在するグラフィクスデータを用いた表示を行う旨を示している。そのため、新たな座標を示す制御情報のみを再生装置に送り込むだけでグラフィクスの表示位置を変えたり、グラフィクスの色彩を変化させるという制御を再生装置に行わせることができる。

制御情報を送り込むだけで、グラフィクス表示を更新することができるので、グラフィクスの頭出し位置と、動画との同期が容易になる。

ここでグラフィクスの表示位置を変化させる場合、グラフィクスそのものの絵柄の変化は必要でないことが多い。人間の目は、動く対象をそれ程鮮明に捉えることができないからである。同じグラフィクスを、高速に位置を変えながら表示するという処理に適した技術が、本発明の本質である。

図面の簡単な説明

図１は、本発明に係る記録媒体の、使用行為についての形態を示す図である。

図２は、BD-ROMの構成を示す図である。

図３は、AVClipがどのように構成されているかを模式的に示す図である。

図４（ａ）は、プレゼンテーショングラフィクスストリームの構成を示す図である。

図４（ｂ）は、機能セグメントを変換することで得られるPESパケットを示す図である。

図５は、様々な種別の機能セグメントにて構成される論理構造を示す図である。

図６は、字幕の表示位置と、Epochとの関係を示す図である。

図７（ａ）は、ODSによるグラフィクスオブジェクトの定義を示す図である。

図７（ｂ）は、PDSのデータ構造を示す図である。

図８（ａ）は、WDSのデータ構造を示す図である。

図８（ｂ）は、PCSのデータ構造で構成される。

図９は、字幕表示を実現するためのDisplay Setの記述例である。

図１０は、DS1におけるWDS、PCSの記述例を示す図である。

図１１は、DS2におけるPCSの記述例を示す図である。

図１２は、DS3におけるPCSの記述例を示す図である。

図１３は、図１０～図１２に示すようなグラフィクスアップデートを実現するにあたっての、オブジェクトバッファにおけるメモリ空間を示す図である。

図１４は、decode_durationの計算アルゴリズムの一例を示す図である。

図１５は、図１４のプログラムのアルゴリズムを図式化したフローチャートである。

図１６（ａ）（ｂ）は、図１４のプログラムのアルゴリズムを図式化したフローチャートである。

図１７（ａ）は、1つのwindowに1つのODSが存在するケースを想定した図である。

図１７（ｂ）（ｃ）は、図１４で引用した各数値の時間的な前後関係を示すタイミングチャートである。

図１８（ａ）は、1つのwindowに2つのODSが存在するケースを想定した図である。

図１８（ｂ）（ｃ）は、図１４で引用した各数値の時間的な前後関係を示すタイミングチャートである。

図１９（ａ）は、2つのwindowのそれぞれに、ODSが1つずつ存在するケースを想定したタイミングチャートである。

図１９（ｂ）は、デコード期間(2)がクリア期間(1)＋書込期間(31)より長くなるケースを示すタイミングチャートである。

図１９（ｃ）は、クリア期間(1)＋書込期間(31)がデコード期間(2)より長くなるケースを示すタイミングチャートである。

図２０は、前方参照を行うPCSを描いた図である。

図２１（ａ）は、グラフィクスが画面内を動き回るという表示効果を実現するEpochを示す図である。

図２１（ｂ）は、DS1～DS8に含まれるPCSの内容と、PTS値とを示す図である。　図２２（ａ）は、DS0内のODSを示す図である。

図２２（ｂ）は、座標(x1,y1)(x2,y2)(x3,y3)･･･(x8,y8)が、ウィンドゥの座標系のどこに存在するかを示す図である。

図２３は、各Display Setに属する各機能セグメントのタイムスタンプに対する設定を示す図である。

図２４は、アップデートの、時間的変遷の具体例を示す図である。

図２５は、Pallet Only Updateを実現する一連のDisplay Setを示す図である。

図２６（ａ）は、DS0内のPDS及び各Display SetのPCSがどのような内容であるかを示す図である。

図２６（ｂ）は、DS0～DS3内のPCSを示す図である。

図２７は、4つのDisplay Setが読み込ませることで実現される、表示効果を現した図である。

図２８は、本発明に係る再生装置の内部構成を示す図である。

図２９は、書込レートRx,Rc,Rd、グラフィクスプレーン８、Coded Data Buffer１３、Object Buffer１５、Composition Buffer１６のサイズを示す図である。

図３０は、再生装置によるパイプライン処理を示すタイミングチャートである。

図３１は、ODSのデコードが、グラフィックスプレーンのクリアより早く終わる場合を想定したパイプライン処理を示すタイミングチャートである。

図３２は、Compositionバッファ１６、Object Buffer１５、Coded Dataバッファ１３、グラフィクスプレーン８における蓄積量の時間的遷移を示すタイミングチャートである。

図３３は、機能セグメントのロード処理の処理手順を示すフローチ

ャートである。

図３４は、多重化の一例を示す図である。

図３５は、DS10が再生装置のCoded Data Buffer１３にロードされる様子を示す図である。

図３６は、通常再生が行われる場合を示す図である。

図３７は、図３６のように通常再生が行われた場合のDS1,10,20のロードを示す図である。

図３８は、Graphical Controller１７の処理手順を示すフローチャートである。

図３９は、Graphical Controller１７の処理手順を示すフローチャートである。

図４０は、Graphical Controller１７の処理手順を示すフローチャートである。

図４１は、第１実施形態に示したPCSが記録されたBD-ROMを製造するための製造工程を示す図である。

発明を実施するための最良の形態

（第１実施形態）

以降、本発明に係る記録媒体の実施形態について説明する。先ず始めに、本発明に係る記録媒体の実施行為のうち、使用行為についての形態を説明する。図１は、本発明に係る記録媒体の、使用行為についての形態を示す図である。図１において、本発明に係る記録媒体は、BD-ROM１００である。このBD-ROM１００は、再生装置２００、テレビ３００、リモコン４００により形成されるホームシアターシステムに、映画作品を供給するという用途に供される。

以上が本発明に係る記録媒体の使用形態についての説明である。

続いて本発明に係る記録媒体の実施行為のうち、生産行為についての形態について説明する。本発明に係る記録媒体は、BD-ROMの応用層に対する改良により実施することができる。図２は、BD-ROMの構成を示す図である。

本図の第４段目にBD-ROMを示し、第３段目にBD-ROM上のトラックを示す。本図のトラックは、BD-ROMの内周から外周にかけて螺旋状に形成されているトラックを、横方向に引き伸ばして描画している。このトラックは、リードイン領域と、ボリューム領域と、リードアウト領域とからなる。本図のボリューム領域は、物理層、ファイルシステム層、応用層というレイヤモデルをもつ。ディレクトリ構造を用いてBD-ROMの応用層フォーマット(アプリケーションフォーマット)を表現すると、図中の第１段目のようになる。本図に示すようにBD-ROMには、ROOTディレクトリの下にBDMVディレクトリがあり、BDMVディレクトリの配下には、AVClipを格納したファイル(XXX.M2TS)、AVClipの管理情報を格納したファイル(XXX.CLPI),AVClipにおける論理的な再生経路(PL)を定義したファイル(YYY.MPLS)が存在する。本図に示すようなアプリケーションフォーマットを作成することにより、本発明に係る記録媒体は生産される。尚、XXX.M2TS、XXX.CLPI,YYY.MPLSといったファイルが、それぞれ複数存在する場合は、BDMVディレクトリの配下に、STREAMディレクトリ、CLIPINFディレクトリ、PLAYLISTディレクトリという3つのディレクトリを設け、STREAMディレクトリにXXX.M2TSと同じ種別のファイルを、CLIPINFディレクトリにXXX.CLPIと同じ種別のファイルを、PLAYLISTディレクトリにYYY.MPLSと同じ種別のファイルを格納することが望ましい。

このアプリケーションフォーマットにおけるAVClip(XXX.M2TS)について説明する。

AVClip(XXX.M2TS)は、MPEG-TS(Transport Stream)形式のデジタルストリームであり、ビデオストリーム、1つ以上のオーディオストリーム、プレゼンテーショングラフィクスストリームを多重化することで得られる。ビデオストリームは映画の動画部分を、オーディオストリームは映画の音声部分を、プレゼンテーショングラフィクスストリームは、映画の字幕をそれぞれ示している。図３は、AVClipがどのように構成されているかを模式的に示す図である。

AVClipは(中段)、複数のビデオフレーム(ピクチャpj1,2,3)からな

るビデオストリーム、複数のオーディオフレームからなるオーディオストリームを(上１段目)、PESパケット列に変換し(上２段目)、更にTSパケットに変換し(上３段目)、同じくプレゼンテーショングラフィクスストリーム(下１段目)を、PESパケット列に変換し(下２段目)、更にTSパケットに変換して(下３段目)、これらを多重化することで構成される。

本図において多重されるプレゼンテーショングラフィクスストリームは1つであるが、BD-ROMが多言語対応型である場合、その言語毎のプレゼンテーショングラフィクスストリームがAVClipに多重される。かかる過程を経て生成されたAVClipは、通常のコンピュータファイル同様、複数のエクステントに分割され、BD-ROM上の領域に記録される。

続いてプレゼンテーショングラフィクスストリームについて説明する。図４（ａ）は、プレゼンテーショングラフィクスストリームの構成を示す図である。第１段目は、AVClipを構成するTSパケット列を示す。第２段目は、グラフィクスストリームを構成するPESパケット列を示す。第２段目におけるPESパケット列は、第１段目におけるTSパケットのうち、所定のPIDをもつTSパケットからペイロードを取り出して、連結することにより構成される。

第３段目は、グラフィクスストリームの構成を示す。グラフィクスストリームは、PCS(Presentation Composition Segment)、WDS(Window Define Segment)、PDS(Palette Difinition Segment)、ODS(Object_Definition_Segment)、END(END of Display Set Segment)と呼ばれる機能セグメントからなる。これらの機能セグメントのうち、PCSは、画面構成セグメントと呼ばれ、WDS,PDS,ODS,ENDは定義セグメントと呼ばれる。PESパケットと機能セグメントとの対応関係は、1対1の関係、1対多の関係である。つまり機能セグメントは、1つのPESパケットに変換されてBD-ROMに記録されるか、又は、フラグメント化され、複数PESパケットに変換されてBD-ROMに記録される。

図４（ｂ）は、機能セグメントを変換することで得られるPESパケ

ットを示す図である。図４（ｂ）に示すようにPESパケットは、パケットヘッダと、ペイロードとからなり、このペイロードが機能セグメント実体にあたる。またパケットヘッダには、この機能セグメントに対応するDTS、PTSが存在する。尚以降の説明では、機能セグメントが格納されるPESパケットのヘッダ内に存在するDTS及びPTSを、機能セグメントのDTS及びPTSとして扱う。

これら様々な種別の機能セグメントは、図５のような論理構造を構築する。図５は、様々な種別の機能セグメントにて構成される論理構造を示す図である。本図は第３段目に機能セグメントを、第２段目にDisplay Setを、第１段目にEpochをそれぞれ示す。

第２段目のDisplay Set(DSと略す)とは、グラフィクスストリームを構成する複数機能セグメントのうち、一画面分のグラフィクスを構成するものの集合をいう。図中の破線は、第３段目の機能セグメントが、どのDSに帰属しているかという帰属関係を示す。

PCS-WDS-PDS-ODS-ENDという一連の機能セグメントが、1つのDSを構成していることがわかる。再生装置は、このDSを構成する複数機能セグメントをBD-ROMから読み出せば、一画面分のグラフィクスを構成することができる。

第１段目のEpochとは、AVClipの再生時間軸上においてメモリ管理の連続性をもっている一つの期間、及び、この期間に割り当てられたデータ群をいう。ここで想定しているメモリとは、一画面分のグラフィクスを格納しておくためのグラフィクスプレーン、伸長された状態のグラフィクスデータを格納しておくためのオブジェクトバッファである。これらについてのメモリ管理に、連続性があるというのは、このEpochにあたる期間を通じてこれらグラフィクスプレーン及びオブジェクトバッファのフラッシュは発生せず、グラフィックスプレーン内のある決められた矩形領域内でのみ、グラフィクスの消去及び再描画が行われることをいう(※ここでフラッシュとは、プレーン及びバッファの格納内容を全部クリアしてしまうことである。)。この矩形領域の縦横の大きさ及び位置は、Epochにあたる期間において、終

始固定されている。グラフィックスプレーンにおいて、この固定化された領域内で、グラフィクスの消去及び再描画を行っている限り、映像とグラフィクスとの同期が保障される。つまりEpochは、映像－グラフィクスの同期の保障が可能な再生時間軸上の一単位ということができる。グラフィックスプレーンにおいて、グラフィクスの消去・再描画を行うべき領域を変更したい場合は、再生時間軸上においてその変更時点を定義し、その変更時点以降を、新たなEpochにせねばならない。この場合、2つのEpochの境界では、映像－グラフィクスの同期は保証されない。

Epochにおける字幕の位置関係にたとえれば、再生時間軸上において、画面上のある決まった矩形領域内に字幕が出現している期間が、Epochということができる。図６は、字幕の表示位置と、Epochとの関係を示す図である。本図では、動画の各ピクチャの絵柄に応じて字幕の位置を変更するという配慮がなされている。つまり5つの字幕「本当は」「ウソだった」「ごめん」「あれから」「3年がたった」のうち、3つの字幕「本当は」「ウソだった」「ごめん」は画面の下側に、「あれから」「3年がたった」は画面の上側に配置されている。これは画面の見易さを考え、画面中の余白にあたる位置に字幕を配置することを意図している。かかる時間的な変動がある場合、AVClipの再生時間軸において、下側の余白に字幕が出現している期間が1つのEpoch1、上側の余白に字幕が出現している期間が別のEpoch2になる。これら2つのEpochは、それぞれ独自の字幕の描画領域をもつことになる。Epoch1では、画面の下側の余白が字幕の描画領域(window1)になる。一方Epoch2では、画面の上側の余白が字幕の描画領域(window2)になる。これらのEpoch1,2において、バッファ・プレーンにおけるメモリ管理の連続性は保証されているので、上述した余白での字幕表示はシームレスに行われる。以上がEpochについての説明である。続いてDisplay Setについて説明する。

図５における破線hk1,2は、第２段目の機能セグメントが、どのEpochに帰属しているかという帰属関係を示す。Epoch

Start, Acquisition Point, Normal Caseという一連のDSは、第1段目のEpochを構成していることがわかる。『Epoch Start』、『Acquisition Point』、『Normal Case』は、DSの類型である。本図におけるAcquisition Point、Normal Caseの順序は、一例にすぎず、どちらが先であってもよい。

『Epoch Start』は、”新表示”という表示効果をもたらすDSであり、新たなEpochの開始を示す。そのためEpoch Startは、次の画面合成に必要な全ての機能セグメントを含んでいる。Epoch Startは、映画作品におけるチャプター等、頭出しがなされることが判明している位置に配置される。

『Acquisition Point』は、”表示リフレッシュ”という表示効果をもたらすDisplay Setであり、先行するEpoch Startと全く同じDisplay Setをいう。Acquisition PointたるDSは、Epochの開始時点ではないが、次の画面合成に必要な全ての機能セグメントを含んでいるので、Acquisition PointたるDSから頭出しを行えば、グラフィックス表示を確実に実現することができる。つまりAcquisition PointたるDSは、Epochの途中からの画面構成を可能するという役割をもつ。

Acquisition PointたるDisplay Setは、頭出し先になり得る位置に組み込まれる。そのような位置には、タイムサーチにより指定され得る位置がある。タイムサーチとは、何分何秒という時間入力をユーザから受け付けて、その時間入力に相当する再生時点から頭出しを行う操作である。かかる時間入力は、10分単位、10秒単位というように、大まかな単位でなされるので、10分間隔の再生位置、10秒間隔の再生位置がタイムサーチにより指定され得る位置になる。このようにタイムサーチにより指定され得る位置にAcquisition Pointを設けておくことにより、タイムサーチ時のグラフィクスストリーム再生を好適に行うことができる。

『Normal Case』は、”表示アップデート”という表示効果をもたらすDSであり、前の画面合成からの差分のみを含む。例えば、あるDSvの字幕は、先行するDSuと同じ内容であるが、画面構成が、この先行

するDSuとは異なる場合、PCSと、ENDのみのDSvを設けてこのDSvをNormal CaseのDSにする。こうすれば、重複するODSを設ける必要はなくなるので、BD-ROMにおける容量削減に寄与することができる。一方、Normal CaseのDSは、差分にすぎないので、Normal Case単独では画面構成は行えない。

続いてDefinition Segment(ODS,WDS,PDS)について説明する。

『Object_Definition_Segment』は、グラフィクスオブジェクトを定義する機能セグメントである。このグラフィクスオブジェクトについて以下説明する。BD-ROMに記録されているAVClipは、ハイビジョン並みの高画質をセールスポイントにしているため、グラフィクスオブジェクトの解像度も、1920×1080画素という高精細な大きさに設定されている。1920×1080という解像度があるので、BD-ROMでは、劇場上映用の字幕の字体、つまり、手書きの味わい深い字体の字幕表示を鮮やかに再現できる。グラフィクスオブジェクトは複数のランレングスデータからなる。ランレングスデータとは、画素値を示すPixel Codeと、画素値の連続長とにより、画素列を表現したデータである。Pixel Codeは、8ビットの値であり、1～255の値をとる。ランレングスデータでは、このPixel CodeによりフルカラーのR16,777,216色から任意の256色を選んで画素の色として設定することができる。尚、字幕として表示される場合、グラフィクスオブジェクトは、透明色の背景に、文字列を配置することで描画せねばならない。

ODSによるグラフィクスオブジェクトの定義は、図７（ａ）に示すようなデータ構造をもってなされる。ODSは、図７（ａ）に示すように自身がODSであることを示す『segment_type』と、ODSのデータ長を示す『segment_length』と、Epochにおいてこの ODSに対応するグラフィクスオブジェクトを一意に識別する『object_id』と、EpochにおけるODSのバージョンを示す『object_version_number』と、『last_in_sequence_flag』と、グラフィクスオブジェクトの一部又は全部である連続バイト長データ『object_data_fragment』とからなる。

『object_id』は、ODSはデコードされて対応するグラフィクスオブジェクトがオブジェクトバッファ上に読み出された場合、このグラフィクスオブジェクトと、オブジェクトバッファにおいてこのグラフィクスオブジェクトが占有している領域とを一意に識別する識別子である。1つ以上のグラフィクスオブジェクトが格納された状態において、オブジェクトバッファのメモリ空間における個々の領域は、このobject_idにより識別されることになる。仮にあるobject_idが2つ以上のDisplay Setに付加されている場合、先行するODSに対応するグラフィクスオブジェクトは、オブジェクトバッファに格納された後、同じobject_idを有する後続のグラフィクスオブジェクトにより上書きされることになる。かようなアップデートは、空き領域の虫食い状の発生や、グラフィクスオブジェクトの離散配置を避けるとの配慮である。その理由は以下の通りである。グラフィクス表示を行うため、オブジェクトバッファ上のグラフィクスオブジェクトは絶えずグラフィックスプレーンに転送されることになる。オブジェクトバッファ内に空き領域が虫食い状に発生したり、また同じobject_idをもったグラフィクスオブジェクトがバラバラにされて離散的に配置されると、グラフィクスオブジェクトを読み出すためのオーバヘッドが発生し、オブジェクトバッファ→グラフィックスプレーンの転送時にあたっての転送効率が落ちる。この効率低下は、グラフィクス一動画の同期表示に悪影響を及ぼしかねない。かかる事情から、オブジェクトバッファ上においてあるobject_idで識別されるグラフィクスオブジェクトが共有している領域は、同じ大きさと同じobject_idとをもったグラフィクスオブジェクトでのみ上書きできるとしている。

グラフィクスオブジェクト上書きにあたって、後続するグラフィクスオブジェクトのサイズは、先行するものと同じサイズでなければならない。先行するグラフィクスオブジェクトにより小さすぎてもいけないし、大き過ぎてもいけない。アップデートにあたってグラフィクスオブジェクトのサイズが同じになるようにグラフィクスオブジェクトを作成しておくというのは、オーサリング担当者にとっての責務

になる。同じIDをもつグラフィクスオブジェクトの縦横サイズを同じにするとの制限は、1つのEpoch内の制限に過ぎない。あるEpochに属するグラフィクスオブジェクトと、次のEpochに属するグラフィクスオブジェクトとでは縦横のサイズが変わっても良い。

『last_in_sequence_flag』、『object_data_fragment』について説明する。PESパケットのペイロードの制限から、字幕を構成する非圧縮グラフィクスが1つのODSでは格納できない場合がある。そのような場合、グラフィクスを分割することにより得られた1部分(フラグメント)がobject_data_fragmentに設定される。1つのグラフィクスオブジェクトを複数ODSで格納する場合、最後のフラグメントを除く全てのフラグメントは同じサイズになる。つまり最後のフラグメントは、それ以前のフラグメントサイズ以下となる。これらフラグメントを格納したODSは、DSにおいて同じ順序で出現する。グラフィクスオブジェクトの最後は、last_in_sequence_flagをもつODSにより指示される。上述したODSのデータ構造は、前のPESパケットからフラグメントを詰めてゆく格納法を前提にしているが、各PESパケットに空きが生じるように、詰めてゆくという格納法であっても良い。以上がODSの説明である。

『Palette Difinition Segment(PDS)』は、色変換用のパレットを定義する情報である。パレットとは、1～255のPixel Codeと、画素値との組合せを示すデータである。ここで画素値は、赤色差成分(Cr値)，青色差成分(Cb値)，輝度成分(Y値)，透明度(T値)から構成される。各ランレングスデータが有するPixel Codeを、パレットに示される画素値に置き換えることで、ランレングスデータは発色されることになる。PDSのデータ構造を図７（ｂ）に示す。図７（ｂ）に示すようにPDSは、自身がPDSであることを示す『segment_type』、PDSのデータ長を示す『segment_length』、このPDSに含まれるパレットを一意に識別する『pallet_id』、EpochにおけるEpochのPDSのバージョンを示す『pallet_version_number』、各エントリーについての情報『pallet_entry』からなる。『pallet_entry』は、各エントリーにお

ける赤色差成分(Cr値),青色差成分(Cb値),輝度成分Y値,透明度(T値)を示す。

続いてWDSについて説明する。

『window_definition_segment』は、グラフィックスプレーンの矩形領域を定義するための機能セグメントである。Epochでは、クリア及び再描画が、グラフィックスプレーンにおけるある矩形領域内で行われている場合のみ、メモリ管理に連続性が生ずることは既に述べている。このグラフィックスプレーンにおける矩形領域は”window”と呼ばれ、このWDSで定義される。図８（ａ）は、WDSのデータ構造を示す図である。本図に示すようにWDSは、グラフィックスプレーンにおいてウィンドゥを一意に識別する『window_id』と、グラフィックスプレーンにおける左上画素の水平位置を示す『window_horizontal_position』と、グラフィックスプレーンにおける左上画素の垂直位置を示す『window_vertical_position』と、グラフィックスプレーンにおけるウィンドゥの横幅を示す『window_width』と、グラフィックスプレーンにおける縦幅を示す『window_height』とを用いて表現される。

window_horizontal_position、window_vertical_position、window_width、window_heightがとりうる値について説明する。これらが想定している座標系は、グラフィックスプレーンの内部領域であり、このグラフィックスプレーンは、縦:video_height、横:video_widthという二次元状の大きさをもつ。

window_horizontal_positionは、グラフィックスプレーンにおける左上画素の水平アドレスであるので、1～video_widthの値をとり、window_vertical_positionは、グラフィックスプレーンにおける左上画素の垂直アドレスであるので1～video_heightの値をとる。

window_widthは、グラフィックスプレーンにおけるウィンドゥの横幅であるので、1～video_width-window_horizontal_positionの値をとり、window_heightは、グラフィックスプレーンにおける縦幅であるので1～video_height-window_vertical_positionの値をとる。

WDSのwindow_horizontal_position、window_vertical_position、window_width、window_heightにより、グラフィックスプレーンの何処にウィンドゥを配置するか、ウィンドゥの大きさをどれだけにするかをEpoch毎に規定することができる。そのため、あるEpochに属するピクチャが表示されている期間において、ピクチャ内の絵柄の邪魔にならないように、ピクチャ上の余白にあたる位置に、ウィンドゥが現れるようオーサリング時に調整しておくことができる。これによりグラフィクスによる字幕表示を見易くすることができる。WDSはEpoch毎に定義可能なので、ピクチャの絵柄に時間的な変動があっても、その変動に応じて、グラフィクスを見易く表示することができる。そのため、結果として、字幕を映像本体に組み込むのと同じレベルにまで映画作品の品質を高めることができる。

続いて『END of Display Set Segment』について説明する。END of Display Set Segmentは、Display Setの伝送の終わりを示す指標であり、Display Setにおける機能セグメントのうち、最後のODSの直後に配置される。この END of Display SetSegmentの内部構成は、自身がEND of Display SetSegmentであることを示す『segment_type』と、当該機能セグメントのデータ長を示す『segment_length』とからなり、これといって説明が必要な構成要素はない。故に図示は省略する。

以上がODS、PDS、WDS、ENDについての説明である。続いてPCSについて説明する。

PCSは、対話的な画面を構成する機能セグメントである。PCSは、図８（ｂ）に示すデータ構造で構成される。本図に示すようにPCSは、『segment_type』と、『segment_length』と、『composition_number』と、『composition_state』と、『pallet_update_flag』と、『pallet_id』と、『composition_object(1)～(m)』とから構成される。

『composition_number』は、0から15までの数値を用いてDisplay Setにおけるグラフィクスアップデートを識別する。どのように識別するかというと、Epochの先頭から本PCSまでにグラフィクスアップデートが存在すれば、それらグラフィクスアップデートを経由する度に、イ

ンクリメントされるというルールでcomposition_numberは設定される。

『composition_state』は、本PCSから始まるDisplay Setが、Normal Caseであるか、ACquisition Pointであるか、Epoch Startであるかを示す。

『pallet_update_flag』は、本PCSにおいてPalletOnly Displey Updateがなされているかどうかを示す。PalletOnly Displey Updateとは、直前のパレットのみを新たなものに切り換えることでなされるアップデートをいう。本PCSでかかるアップデートがなされれば、本フィールドは”1”に設定される。

『pallet_id』は、PalletOnly Displey Updateに用いられるべきパレットを示す。

『composition_object(1)・・・(n)』は、このPCSが属するDisplay Setにおける画面構成を実現するための制御情報である。図８（ｂ）の破線wd1は、任意のcomposition_object(i)の内部構成をクローズアップしている。この破線wd1に示すように、composition_object(i)は、『object_id_ref』、『window_id_ref』、『object_cropped_flag』、『object_horizontal_position』、『object_vertical_position』、『cropping_rectangle情報(1)(2)・・・・(n)』からなる。

『object_id_ref』は、グラフィクスオブジェクト識別子(object_id)の参照値である。この参照値は、composition_object(i)に対応する画面構成を実現するにあたって、用いるべきグラフィクスオブジェクトの識別子を意味する。

『window_id_ref』は、ウィンドゥ識別子(window_id)の参照値である。この参照値は、composition_object(i)に対応する画面構成を実現するにあたって、どのウィンドゥに、グラフィクスオブジェクトを表示させるべきかを示す。

『object_cropped_flag』は、オブジェクトバッファにおいてクロップされたグラフィクスオブジェクトを表示するか、グラフィクスオブジェクトを非表示とするかを切り換えるフラグである。”1”と設

定された場合、オブジェクトバッファにおいてクロップされたグラフィクスオブジェクトが表示され、"0"と設定された場合、グラフィクスオブジェクトは非表示となる。

『object_horizontal_position』は、グラフィックスプレーンにおけるグラフィクスオブジェクトの左上画素の水平位置を示す。

『object_vertical_position』は、グラフィックスプレーンにおける左上画素の垂直位置を示す。

『cropping_rectangle情報(1)(2)・・・・・(n)』は、『object_cropped_flag』が1に設定されている場合に有効となる情報要素である。破線wd2は、任意のcropping_rectangle情報(i)の内部構成をクローズアップしている。この破線に示すようにcropping_rectangle情報(i)は、『object_cropping_horizontal_position』、『object_cropping_vertical_address』、『object_cropping_width』、『object_cropping_height』からなる。

『object_cropping_horizontal_position』は、グラフィックスプレーンにおけるクロップ矩形の左上画素の水平位置を示す。クロップ矩形は、グラフィクスオブジェクトの一部を切り出すための枠であり、ETSI EN 300 743標準規格における"Region"に相当する。

『object_cropping_vertical_address』は、グラフィックスプレーンにおけるクロップ矩形の左上画素の垂直位置を示す。

『object_cropping_width』は、グラフィックスプレーンにおけるクロップ矩形の横幅を示す。

『object_cropping_height』は、グラフィックスプレーンにおけるクロップ矩形の縦幅を示す。

以上がPCSのデータ構造である。続いてPCSの具体的な記述について説明する。この具体例は、図６に示した字幕の表示、つまり動画の再生進行に伴い、三回のグラフィックスプレーンへの書き込みで『ほんとは』『ウソだった』『ごめん』というように徐々に表示させるというものである。図９は、かかる字幕表示を実現するための記述例であ

る。本図におけるEpochは、DS1(Epoch Start)、DS2(Normal Case)、DS3(Normal Case)を有する。DS1は、字幕の表示枠となるwindowを定義するWDS、台詞『ほんとは　ウソだった　ごめん』を表すODS、1つ目のPCSを備える。DS2(Normal Case)は、2つ目のPCSを有する。DS3(Normal Case)は3つ目のPCSを有する。

次に個々のPCSをどのように記述するかについて説明する。Display Setに属するWDS、PCSの記述例を図１０～図１２に示す。図１０は、DS1におけるPCSの記述例を示す図である。

図１０において、WDSのwindow_horizontal_position、window_vertical_positionは、グラフィックスプレーンにおけるウィンドゥの左上座標LP1を、window_width,window_heightは、ウィンドゥの表示枠の横幅、縦幅を示す。

図１０におけるクロップ情報のobject_cropping_horizontal_position,object_cropping_vertical_positionは、オブジェクトバッファにおけるグラフィクスオブジェクトの左上座標を原点とした座標系においてクロップ範囲の基準ST1を示している。そして基準点からobject_cropping_width、object_cropping_heightに示される範囲(図中の太枠部分)がクロップ範囲になる。クロップされたグラフィクスオブジェクトは、グラフィックスプレーンの座標系においてobject_horizontal_position,object_vertical_positionを基準点(左上)とした破線の範囲cp1に配置される。こうすることにより、『本当は』がグラフィックスプレーンにおけるウィンドゥ内に書き込まれる。これにより字幕『本当は』は動画像と合成され表示される。

図１１（ａ）は、DS2におけるPCSの記述例を示す図である。本図におけるWDSの記述は、図１０と同じなので説明を省略する。クロップ情報の記述は、図１０と異なる。図１１（ａ）におけるクロップ情報のobject_cropping_horizontal_position,object_cropping_vertical_position,は、オブジェクトバッファ上の字幕『本当はウソだった。

ごめん』のうち、『ウソだった』の左上座標を示し、object_cropping_height,object_cropping_widthは、『ウソだった』の横幅、縦幅を示す。こうすることにより、『ウソだった』がグラフィックスプレーンにおけるウィンドゥ内に書き込まれる。これにより字幕『ウソだった』は動画像と合成され表示される。

図１１（ｂ）は、DS3におけるPCSの記述例を示す図である。本図におけるWDSの記述は、図１０と同じなので説明を省略する。クロップ情報の記述は、図１０と異なる。図１１（ｂ）におけるクロップ情報のobject_cropping_horizontal_position,object_cropping_vertical_position,は、オブジェクトバッファ上の字幕『本当はウソだった。ごめん』のうち、『ごめん』の左上座標を示し、object_cropping_height,object_cropping_widthは、『ごめん』の横幅、縦幅を示す。こうすることにより、『ごめん』がグラフィックスプレーンにおけるウィンドゥ内に書き込まれる。これにより字幕『ごめん』は動画像と合成され表示される。　図１３は、図１０～図１２に示すようなグラフィクスアップデートを実現するにあたっての、オブジェクトバッファにおけるメモリ空間を示す図である。本図に示すようにオブジェクトバッファは、縦幅、横幅、位置が固定された格納領域A,B,C,Dが4つあり、このうち格納領域Aに、図１１に示した字幕が格納される。これら4つの格納領域A,B,C,Dのそれぞれには、格納されているグラフィクスオブジェクトに対応するobject_idにより識別される。つまりobject_id=1により格納領域Aが、object_id=2により格納領域Bが、object_id=3により格納領域Cがそれぞれ識別されるのである。グラフィックスプレーンへの転送効率を維持するため、これら格納領域の縦幅、横幅は終始固定されている。同じobject_idをもったグラフィクスオブジェクトがデコードにより新たに得られれば、それらの格納領域はその新たに得られたグラフィクスオブジェクトにより上書きされる。例えば、図１０に示した字幕と同じ位置に、同じ大きさで字幕を表示させたい場合、後続するDisplay Setにおいて、

同じobject_idを付加したODSを設ければよい。このように同じobject_idを付加するだけで、オブジェクトバッファ上に存在するグラフィクスオブジェクトは、新たなグラフィクスオブジェクトで上書きされ、その新たなグラフィクスオブジェクトは、先のグラフィクスオブジェクトと同じ大きさ・同じ位置で表示されることになる。

表示効果実現にあたっての制約について説明する。なめらかな表示効果の実現には、ウィンドゥクリアと、ウィンドゥ再描画とが必要になる。ウィンドゥクリアと、ウィンドゥ再描画とをビデオフレームの時間間隔で実現する場合、オブジェクトバッファと、グラフィックスプレーンとの間の転送レートはどれだけ必要であろうか。

ここでウィンドゥをどれだけの大きさとするかの制限について検討する。オブジェクトバッファーグラフィックスプレーン間の転送レートをRcとすると、ワーストケースでは、この転送レートRcでウィンドゥクリアと、ウィンドゥ再描画とを行わねばならない。そうするとウィンドゥクリア、ウィンドゥ再描画のそれぞれをRcの半分の転送レート(Rc/2)で実現せねばならない。

これらウィンドゥクリアーウィンドゥ再描画をビデオフレームと同期させるには、

ウィンドゥサイズ×フレームレート≒Rc/2

を満たす必要がある。このフレームレートが29.97であるなら、

Rcは、ウィンドゥサイズ×2×29.97になる。

字幕の表示にあたっては、グラフィックスプレーン全体に対し、最低でも25%～33%程度の大きさが必要となる。ここでグラフィックスプレーンの総画素数は1920×1080であり、一画素当たりのインデックスのビット長を8ビットとすると、グラフィックスプレーンの総容量は2Mバイト(≒1920×1080×8)になる。

この総容量の1/4をウィンドゥサイズとすると、500kバイト(＝2Mバイト/4)になる。これを上述した式に代入してRcを導けば、Rcは256Mbps(500Kバイト×2×29.97)と算出することができる。

この25%～33%という大きさであれば、256Mbpsという転送レートで

字幕の表示を行っている限り、如何なる表示効果を実現する場合であっても、動画との同期を維持することができる。

仮に、ウィンドゥクリア及び再描画のレートがビデオフレームのフレームレートの1/2,1/4でよいなら、Rcがたとえ同じであってもウィンドゥサイズを2倍,4倍にすることができる。以上がwindowの大きさについての説明である。続いて、windowの位置、範囲について説明する。上述したように、Epochにおいてウィンドゥの位置、範囲は一貫している。

Epochにおいてウィンドゥの位置、範囲を一貫させておくのは以下の理由による。ウィンドゥの位置・範囲を変えれば、グラフィックスプレーンに対する書込先アドレスを変えねばならず、オーバーヘッドが発生するので、かかるオーバーヘッドによりオブジェクトバッファからグラフィックスプレーンへの転送レートが低下するからである。

ウィンドゥには、グラフィクスオブジェクトの個数が制限されている。この個数制限は、デコードされたグラフィクスオブジェクトの転送にあたってのオーバーヘッドを低減する目的で設けられている。ここでのオーバーヘッドは、グラフィクスオブジェクトのエッジ部分のアドレスを設定する際に発生する。そうすれば、エッジの部分が多く存在する程、オーバーヘッドの発生回数が多くなる。

ウィンドゥにおけるグラフィクスオブジェクトの数に制限がないと、グラフィクス転送にあたって発生するオーバーヘッド数が未知数になり、転送負荷の増減が激しくなる。一方、ウィンドゥにおけるグラフィクスの個数が2つまででであると、最悪4つのオーバーヘッドが発生すると見込んで転送レートを設定すればよいので、ミニマムスタンダードたる転送レートを数値化し易くなる。

以上がウィンドゥについての説明である。続いてこれらPCS、ODSを有したDisplay Setが、AVClipの再生時間軸上にどのように割り当てられるかについて説明する。Epochは、再生時間軸上においてメモリ管理が連続する期間であり、Epochは1つ以上のDisplay Setから構成されるので、Display SetをどうやってAVClipの再生時間軸に割り当

てるかが問題になる。ここでAVClipの再生時間軸とは、AVClipに多重されたビデオストリームを構成する個々のピクチャデータのデコードタイミング、再生タイミングを規定するために想定される時間軸をいう。この再生時間軸においてデコードタイミング、再生タイミングは、90KHzの時間精度で表現される。Display Set内のPCS、ODSに付加されたDTS、PTSは、この再生時間軸において同期制御を実現すべきタイミングを示す。このPCS、ODSに付加されたDTS、PTSを用いて同期制御を行うことが、再生時間軸へのDisplay Setの割り当てである。

先ず、ODSに付加されたDTS、PTSにより、どのような同期制御がなされるかについて説明する。

DTSは、ODSのデコードを開始すべき時間を90KHzの時間精度で示しており、PTSはデコード終了時刻を示す。

ODSのデコードは、瞬時には完了せず、時間的な長さをもっている。このデコード期間の開始点・終了点を明らかにしたいとの要望から、ODSについてのDTS、PTSはデコード開始時刻、デコード終了時刻を示している。

PTSの値はデッドラインであるので、PTSに示される時刻までにODSのデコードがなされて、非圧縮状態のグラフィックオブジェクトが、再生装置上のオブジェクトバッファに得られなければならない。

DSnに属する任意のODSjのデコード開始時刻は、90KHzの時間精度でDTS(DSn[ODSj])に示されるので、これにデコードを要する最長時間を加えた時刻が、Display SetのODSjのデコード終了時刻になる。

ODSjのサイズを"SIZE(DSn[ODSj])"、ODSのデコードレートを"Rd"とすると、デコードに要する最長時間(秒)は、"SIZE(DSn[ODSj])//Rd"になる。ここで"//"は、小数点以下切り上げの割り算を示す演算子である。

この最長時間を90KHzの時間精度に変換し、ODSjのDTSに加算することにより、PTSで示されるべきデコード終了時刻(90KHz)は算出される。

DSnに属するODSjのPTSを、数式で表すと、以下の式のようになる。

PTS(DS[ODSj])=DTS(DSn[ODSj])+90,000×(SIZE(DSn[ODSj])//Rd)

そして互いに隣接する2つのODS(ODSj,ODSj+1)との間では、以下の関係を満たす必要がある。

PTS(DSn[ODSj])≦DTS(DSn[ODSj+1])

またDSnに属するENDは、DSnの終わりを示すものだから、DSnに属する最後のODS(ODSlast)のデコード終了時刻を示せばよい。このデコード終了時刻は、ODS2(ODSlast)のPTS(PTS(DSn[ODSlast]))に示されているので、ENDのPTSは、以下の式に示される値に設定される。

PTS(DSn[END]) = PTS(DSn[ODSlast])

続いてPCSのDTS、PTSの設定について説明する。

PCSのDTS値は、DSnにおける最初のODS(ODS1)のデコード開始時点か、それ以前の時点を示す。何故ならPCSは、DSnにおける最初のODS(ODS1)のデコード開始時点(DTS(DSn[ODS1]))、及び、DSnにおける最初のPDS(PDS1)が有効になる時点(PTS(DSn[PDS1]))と同時か、又はそれ以前に、再生装置上のバッファにロードされねばならないからである。よって以下の式の関係を満たす値に、設定されねばならない。

]DTS(DSn[PCS])≦DTS(DSn[ODS1])

DTS(DSn[PCS])≦PTS(DSn[PDS1])

そしてDSnにおけるPCSのPTSは、以下の数式から算出される。

PTS(DSn[PCS])≧DTS(DSn[PCS])+decodeduration(DSn)

ここでdecodeduration(DSn)は、PCSのアップデートに用いられる全グラフィクスオブジェクトのデコード及び表示に要する時間である。このデコード時間は、固定値ではない。しかし各再生装置の状態や再生装置の実装により変動するものでもない。本DSn.PCSnの画面合成に用いられるオブジェクトをDSn.PCSn.OBJ[j]とした場合、decodeduration(DSn)は、ウィンドゥクリアに要する時間(i)、DSn.PCSn.OBJのデコード期間(ii)、DSn.PCSn.OBJの書き込みに要する時間(iii)により変動を受ける値になる。Rd,Rcさえ予め定められていれば、どのような実装の再生装置であっても、同じ値になる。そのためオーサリング時にあたっては、これらの期間の長さを算出して、こ

の値からPTSを計算している。

decode_durationの計算は、図１４のプログラムに基づき行われる。また図１５、図１６（ａ）（ｂ）は、このプログラムのアルゴリズムを図式化したフローチャートである。以降、これらの図を参照しながらdecode_durationの算出について説明する。図１５のフローチャートにおいて、先ず初めに、PLANEINITIALIZATINTIME関数を呼び出し、戻り値をdecode_durationに加算する(図１５のステップＳ１)。PLANEINITIALIZATINTIME関数(図１６（ａ）)は、Display Setを表示するにあたってグラフィックスプレーンの初期化に要する時間を求める関数を呼出すための関数であり、図１５のステップＳ１では、DSn,DSn.PCS.OBJ[0],decode_durtationを引数に設定して、この関数を呼び出す。

続いて図１６（ａ）を参照しながらPLANEINITIALIZATINTIME関数について説明する。図１６（ａ）においてinitialize_durationとは、PLANEINITIALIZATINTIME関数の戻り値を示す変数である。

図１６のステップＳ２は、DSnのPCSにおけるcomposition_stateがEpoch Startかどうかにより、処理を切り換えるif文である。もしcomposition_stateがEpoch Startであるなら(図１４のDSn.PCS.composition_state==EPOCH_START、ステップＳ２＝Yes)、グラフィックスプレーンのクリアに要する時間をinitialize_durationに設定する(ステップＳ３)。

オブジェクトバッファーグラフィックスプレーン間の転送レートRcを上述したように256,000,000とし、グラフィックスプレーンの総サイズをvideo_width*video_heightとすると、クリアに要する時間(秒)は、「video_width*video_height//256,000,000」になる。これに、90,000Hzを乗じPTSの時間精度で表現すれば、グラフィックスプレーンのクリアに要する時間は「90000×video_width*video_height//256,000,000」になる。この時間を、initialize_durationに加えて、initialize_durationを戻り値としてリターンする。

もしcomposition_stateがEpoch Startでないなら(ステップＳ２＝No)、WDSにて定義されるwindow[i]のクリアに要する時間をinitialize_durationに加えるという処理を全てのwindowについて繰り返す(ステップＳ４)。オブジェクトバッファーグラフィックスプレーン間の転送レートRcを上述したように256,000,000とし、WDSに属するウィンドゥ[i]の総サイズをΣSIZE(WDS.WIN[i])とすると、クリアに要する時間(秒)は、「ΣSIZE(WDS.WIN[i])//256,000,000」になる。これに、90,000Hzを乗じPTSの時間精度で表現すれば、WDSに属する全ウィンドゥのクリアに要する時間は「90000×ΣSIZE(WDS.WIN[i]))//256,000,000」になる。この時間を、initialize_durationに加えて、initialize_durationを戻り値としてリターンする。以上がPLANEINITIALIZATINTIME関数である。

図１５のステップＳ５は、DSnに属するグラフィクスオブジェクトの数が2であるか、1であるかで処理を切り換えるものであり(図１４のif(DSn.PCS.num_of_object==2),if(DSn.PCS.num_of_object==1))、もし”1”であるなら(ステップＳ５＝1)、グラフィクスオブジェクトのデコードを行うための待ち時間をdecode_durationに加える(ステップＳ６)。この待ち時間の算出は、WAIT関数の呼出(図１４のdecode_duration +=WAIT(DSn,DSn.PCS.OBJ[0],decode_duration))で実現される。この関数呼出にあたっての引数はDSn,DSn.PCS.OBJ[0],decode_durationであり、待ち時間を示すwait_durationが戻り値として返される。

図１６(ｂ)は、WAIT関数の処理手順を示すフローチャートである。

このWAIT関数においてcurrent_durationとは、呼出元のdecode_durationが設定される。object_define_ready_timeは、Display Setのグラフィクスオブジェクト[i]のPTSが設定される変数である。

current_timeとは、DSnのPCSのDTSに、current_durationを足した値が設定される変数である。このcurrent_timeよりobject_define_ready_timeが大きい場合(ステップＳ７がYes、

if(current_time < object_define_ready_time))、戻り値たるwait_durationは、object_define_ready_timeとcurrent_timeとの差分が設定されることになる(ステップＳ８、wait_duration += object_define_ready_time - current_time)。以上がWait関数である。ステップＳ６においてdecode_durationには、このwait関数の戻り値と、OBJ[0]をwindowに描画するのに必要な時間を足し合わせた時間(90,000*(SIZE(DSn.WDS.WIN[0]))//256,000,000)が設定されることになる(ステップＳ９)。

以上はDSnのグラフィクスオブジェクトが1つである場合の処理である。図１５のステップＳ５は、グラフィクスオブジェクトの数が2であるかどうかを判定している。DSnのグラフィクスオブジェクトが2以上であれば(図１４のif(DSn.PCS.num_of_object==2))、DSn,DSn.PCS.OBJ[0],decode_durationを引数として、wait関数を呼び出し、その戻り値をdecode_durationに加算する(ステップＳ１０)。

続くステップＳ１１は、DSnのOBJ[0]が属するwindowが、グラフィクスオブジェクト[1]が属するwindowと同一かどうかの判定であり(if(DSn.PCS.OBJ[0].window_id == DSn.PCS.OBJ[1].window_id)、もし同一であるなら、DSn,DSn.PCS.OBJ[1],decode_durationを引数にしてwait関数を呼び出し、その戻り値wait_durationを、decode_durationに加算する(ステップＳ１２)。OBJ[0]が属するwindowの再描画に必要な時間(90,000*(SIZE(DSn.WDS.OBJ[0].window_id)//256,000,000)をdecode_durationに加える(ステップＳ１３)。

もし属するwindowが異なるなら(ステップＳ１１で”異なる”)、OBJ[0]が属するwindowの再描画に必要な時間(90,000*(SIZE(DSn.WDS.OBJ[0].window_id)//256,000,000)をdecode_durationに加える(ステップＳ１５)。DSn,DSn.PCS.OBJ[1],decode_durationを引数にしてwait関数を呼び出し、その戻り値wait_durationを、decode_durationに加算する(ステップＳ１６)。その後、OBJ[1]が属するwindowの再描画に必要な時

間(90,000*(SIZE(DSn.WDS.OBJ[1].window_id)//256,000,000)をdecode_durationに加える(ステップＳ１７)。

以上のアルゴリズムにより、Decode_Durationは算出される。このPCSのPTSが、具体的にどのように設定されるかについて説明する。

図１７（ａ）は、1つのwindowに1つのODSが存在するケースを想定した図である。図１７（ｂ）（ｃ）は、図１４で引用した各数値の時間的な前後関係を示すタイミングチャートである。本チャートには3つの段があり、これらの段のうち、”グラフィックスプレーンアクセス”の段、”ODSデコード”の段は、再生時にパラレルになされる2つの処理を示す。上述したアルゴリズムは、これらの2つの処理のパラレル実行を想定している。

グラフィックスプレーンアクセスは、クリア期間(1)、書き込み期間(3)からなる。このクリア期間(1)は、グラフィックスプレーン全体のクリアに要する期間(90,000×(グラフィックスプレーンのサイズ//256,000,000))、グラフィックスプレーンにおける全windowのクリアに要する時間(Σ(90,000×(window[i]のサイズ//256,000,000)))のどちらかを意味する。

書き込み期間(3)は、window全体描画に要する期間(90,000×(windowサイズ//256,000,000))を意味する。

一方、デコード期間(2)は、ODSのDTSからPTSまでに示される期間を意味する。　これらクリア期間(1)～書き込み期間(3)は、クリアすべき範囲、デコードすべきODSのサイズ、グラフィックスプレーンに書き込むべきグラフィクスオブジェクトのサイズにより変化し得る。本図では、説明の簡略化のため、ODSのデコード期間(2)の始点は、クリア期間(1)の始点と同一であると仮定している。

図１７（ｂ）はデコード期間(2)が長くなるケースであり、Decode_Durationはデコード期間(2)＋書き込み期間(3)になる。

図１７（ｃ）は、クリア期間(1)が長くなるケースであり、Decode_Durationはクリア期間(1)＋書き込み期間(3)の期間がDecode_Durationになる。

図１８（ａ）～（ｃ）は、1つのwindowに2つのODSが存在するケースを想定した図である。本図（ｂ）（ｃ）におけるデコード期間(2)は、2つのグラフィクスのデコードに要する期間の総和を意味する。グラフィクス書込期間(3)も、2つのグラフィクスをグラフィックスプレーンに書き込む期間の総和を意味する。ODSが2つになっているものの、図１７と同様に考えればDecode_Durationを算出することができる。2つのODSをデコードするためのデコード期間(2)が長い場合は、図１８（ｂ）に示すようにDecode_Durationはデコード期間(2)＋書き込み期間(3)に算出されることになる。

クリア期間(1)が長い場合は、図１８（ｃ）に示すように、Decode_Durationはクリア期間(1)＋書き込み期間(3)となる。

図１９（ａ）は、2つのwindowのそれぞれに、ODSが1つずつ存在するケースを想定している。この場合でもクリア期間(1)が、2つのODSをデコードするための総デコード期間(2)より長い場合、Decode_Durationはクリア期間(1)＋書き込み期間(3)になる。問題は、クリア期間(1)がデコード期間(2)より短くなるケースである。この場合デコード期間(2)の経過を待たずに、1つ目のwindowへの書き込みは可能になる。そのためクリア期間(1)＋書き込み期間(3)、デコード期間(2)＋書き込み期間(3)の長さにはならない。ここで1つ目のODSのデコードに要する期間を書込期間(31)、2つ目のODSのデコードに要する期間を書込期間(32)とする。図１９（ｂ）は、デコード期間(2)がクリア期間(1)＋書込期間(31)より長くなるケースを示す。この場合Decode_Durationは、デコード期間(2)＋書込期間(32)になる。

図１９（ｃ）は、クリア期間(1)＋書込期間(31)がデコード期間(2)より長くなるケースを示す。この場合Decode_Durationはクリア期間(1)＋書込期間(31)＋書込期間(32)になる。

グラフィックスプレーンのサイズは、プレーヤモデルから予め判明しており、またwindowのサイズ、ODSのサイズ、個数もオーサリングの段階で判明しているので、これらからDecode_Durationがクリア期間(1)＋書き込み期間(3)、デコード期間(2)＋書き込み期間(3)、デコ

ード期間(2)＋書込期間(32)、クリア期間(1)＋書込期間(31)＋書込期間(32)のどれかになる。こうしたDecode_Duration算出を基にPCSのPTSを設定すれば、ピクチャデータとの同期表示を高い時間精度で実現することができる。このような高精度な同期制御は、windowを定義し、クリア・再描画する範囲を、このwindowに限定することで成り立っているので、オーサリングの環境に、このwindowの概念を導入したことの意義は大きい。

続いてDSnにおけるWDSのDTS、PTSの設定について説明する。WDSのDTSは、以下の数式を満たすように設定すればよい。

DTS(DSn[WDS])≧DTS(DSn[PCS])

一方、DSnにおけるWDSのPTSは、グラフィックスプレーンに対する書き込みを開始すべきデッドラインを示す。グラフィックスプレーンへの書き込みは、ウィンドゥだけで足りるので、PCSのPTSに示される時刻から、WDSの書き込みに要する期間を差し引けば、グラフィックスプレーンへの書き込みを開始すべき時刻は定まる。WDSの総サイズをΣSIZE(WDS.WIN[i])とすれば、これのクリア及び再描画に要する時間は、『ΣSIZE(WDS.WIN[i])//256,000,000』になる。そして、これを90.000KHzの時間精度で表現すると『90000×Σ SIZE(WDS.WIN[i])//256,000,000』になる。

故に以下の数式から、WDSのPTSを算出すればよい。

PTS(DSn[WDS])=PTS(DSn[PCS])-90000×Σ SIZE(WDS.WIN[i])//256,000,000

このWDSに示されるPTSはデッドラインなので、これより早い時点からグラフィックスプレーンの書き込みを開始してもよい。つまり図１９に示すように、2つのウィンドゥのうち、1つのウィンドゥに表示させるべきODSのデコードが完了したなら、その時点からデコードにより得られたグラフィクスオブジェクトの書き込みを開始してもよい。

このようにWDSに付加されたDTS、PTSを用いてAVClipの再生時間軸の任意の時点に、ウィンドゥを割り当てることができる。

以上がPCS、WDSのDTS、PTSの説明である。

続いてPCSによる前方参照について説明する。前方参照とは、前もって再生装置のメモリに読み込まれたグラフィクスオブジェクトを参照した再生制御をいう。図２０は、前方参照を行うPCSを描いた図である。本図では、DSn、DSn+1、DSn+2という3つのDisplay Setが描かれている。この中でDSn+1、DSn+2に属するPCSが「前方参照を行うPCS」である。本図の矢印yy1,yy2は、PCSにおける前方参照を象徴的に示す。かかる前方参照において参照先になっているグラフィクスオブジェクトは、DSn内のODS#1～#vにより定義されるグラフィクスオブジェクトである。ここでDSnは、ODS#1～#u、ODS#u+1～#vという複数のグラフィクスオブジェクトを含む。このうちDSn内のPCSにより参照されているのは、ODS#1～#uであり、ODS#u+1～#vは参照されていない。つまりDSn内の非参照グラフィクスオブジェクトが前方参照されているのである。

以上のような前方参照では、PCSにより参照されているグラフィクスオブジェクトが前もって再生装置に読み込まれているので、同じDisplay Setに属するODSのデコードを待つ必要が無い。PCSを新たに送り込むだけで、そのPCSに基づく表示制御を再生装置に実行させることができる。PCSの読み込みに即応した表示制御が可能であるため、既に表示されているグラフィクスを、新たなグラフィクスを用いて更新するというグラフィクス更新を短い時間間隔で実行することができる。そして短い時間間隔でのグラフィクス更新を繰り返せば、動画像の再生進行と共にグラフィクスを滑らかに動かすという表示制御が可能になる。前方参照型のPCSにおける2つの記述例について説明する。

先ず初めに、画面上でグラフィクスが動きまわるような表示効果を実現するにあたってのPCSの記述について説明する。図２１（ａ）は、かかる表示効果を実現するEpochを示す図である。本Epochは、9つのDisplay Set(DS0～DS8)からなり、先頭のDS0は、PCS、PDS、ODSを含む。DS0内のODSは、図２２（ａ）に示すような字幕「心が揺れる」を

定義するものであり、object_id=1が付されている。後続するDS1～DS8は、PCSのみを含む。

図２１（ｂ）は、DS1～DS8に含まれるPCSの内容と、PTS値とを示す図である。本図に示すようにDS1～DS8に属するPCS内の内容は、object_id_ref=1のグラフィクスオブジェクトを座標(x1,y1)(x2,y2)(x3,y3)･･･(x8,y8)に表示するよう、object_horizontal_position,object_vertivcal_positionの位置決めがなされている。これら座標(x1,y1)(x2,y2)(x3,y3)･･･(x8,y8)が、ウィンドゥの座標系のどこに存在するかを、具体的に示したのが図２２（ｂ）である。図２２（ｂ）において座標(x1,y1)(x2,y2)(x3,y3)･･･(x8,y8)は、波形を表す曲線上に存在していることがわかる。これら(x1,y1)(x2,y2)(x3,y3)･･･(x8,y8)という座標値は、WDSにて定義されるウィンドゥ内に、グラフィクスが収まるように規定されている。ウィンドゥ内に、グラフィクスが収まらないと、ビデオストリームにおける動画表示との同期を確立することができないからである。

図２１（ｂ）におけるPCSのPTS値は、座標(x1,y1)(x2,y2)(x3,y3)･･･(x8,y8)に対する表示を、再生時間軸上の時点t1,t2,t3････t8のタイミングで命じるものである。

以上のように、PCSの内容が規定されたDS0～DS8を順次再生装置に読み込んでゆくことにより、グラフィクスは、図２３のような表示効果をなす。動画ストリームの再生時間軸における時点t1において、再生装置にはDS1が読み込まれるので、ウィンドゥ内の座標(x1,y1)にグラフィクスが表示される。時点t4において再生装置にはDS4が読み込まれるので座標(x4,y4)にグラフィクスが表示されることになる。一方時点t6では、DS6が読み込まれるので座標(x6,y6)に、時点t8ではDS8が読み込まれるので座標(x8,y8)にグラフィクスが表示されることになる。これらの座標(x1,y1),(x4,y4),(x6,y6),(x8,y8)は、波形の曲線上に存在するので、かかる座標に表示位置の位置決めがなされたPCSにより、グラフィクスは、図２４に示すように波形の軌道を描くことになる。

以上のように
object_horizontal_position,object_vertivcal_positionが規定されたPCSを含むDisplay Setを順次再生装置に読み込ませれば、再生装置はBD-ROMから読み出されてくるDS1、DS2、DS3内のPCSに従い、既にオブジェクトバッファに格納されているグラフィクスオブジェクトを、表示させることができる。制御情報の送り込みのみで、グラフィクスの位置を変化させることができるので、グラフィクスを目まぐるしく動かすという制御に適している。

個々のDisplay Setの読み出し時において、それまで参照していたPCSを、新たなPCSで上書きするとの動作を行えば、たとえ読み出すべきPCSが何十個、何百個あったとしても、そのうち最も新しいもののみ、再生装置に格納しておけばよい。滑らかな動きを実現するため、表示座標を示すPCSが、何十個、何百個必要であっても、そのうち最新のもののみ、再生装置に常駐させておけばよいので、再生装置におけるメモリ占有量を必要最低限にすることができる。

続いてPCSの第２の記述例としてPallet Only Updateについて説明する。図２５は、Pallet Only Updateを実現する一連のDisplay Setを示す図である。本図の第１段目は、Pallet Only Updateを実現する一連のDisplay Set(DS0、DS1、DS2、DS3)を示す。このうちDS0は、PCSと、3つのPDS0,1,2,3と、ODSとを含む。以降のDS2、DS3、DS4は、PCSのみからなる。

DS0内のODSは、引き出し線hv1に示すように、文字列「忘れない」を定義するものである。これら文字「忘」「れ」「な」「い」は、それぞれ複数のRun Length(図中の白抜きの四角形)から構成される。このうち文字「忘」を構成するRun Lengthは、PixelCode0をもっているものとする。文字「れ」,「な」,「い」を構成するRun Lengthは、PixelCode1,PixelCode2,PixelCode3をもっているものとする。

図２６（ａ）は、DS0内のPDS及び各Display SetのPCSがどのような内容であるかを示す図である。

4つのPDSのうち、1つ目のPDS0は(第1段目)、pallet_id=0が付加さ

れており、PixelCode1に赤色を、PixelCode2以降に白色を割り当てるものである。

2つ目のPDS1は(第2段目)、pallet_id=1が付加されており、1,2のPixelCodeに赤色を、3以降のPixelCodeに白色を割り当てるものである。

3つ目のPDS2は(第3段目)、pallet_id=2が付加されており、1,2,3のPixelCodeに赤色を、4のPixelCodeに白色を割り当てるものである。

4つ目のPDS3は(第4段目)、pallet_id=3が付加されており、1～4のPixelCodeに赤色を割り当てるものである。

図２６（ｂ）は、DS0～DS3内のPCSを示す図である。

本図においてDS0内のPCSは(第1段目)、object_id=1のグラフィクスオブジェクトを参照した表示制御を行う旨を示す。そしてPallet_only_updateが0に設定され、Pallet_idが0に設定されている。これによりDS0内のODSは、Pallet_id=0で指示されるPDSを用いて、表示されることになる。

DS1内のPCSは(第2段目)、object_id=1のグラフィクスオブジェクトを参照した表示制御を行う旨を示す。そしてPallet_only_updateが1に設定され、Pallet_idが1に設定されている。これは、DS1の読み込み時において、Pallet_id=1で指示されるPDSを用いて、グラフィクス表示の更新を行う旨を示す。

DS2内のPCSは(第3段目)、object_id=1のグラフィクスオブジェクトを参照した表示制御を行う旨を示す。そしてPallet_only_updateが1に設定され、Pallet_idが2に設定されている。これは、DS2の読み込み時において、Pallet_id=2で指示されるPDSを用いて、グラフィクス表示の更新を行う旨を示す。]

DS3内のPCSは(第4段目)、object_id=1のグラフィクスオブジェクトを参照した表示制御を行う旨を示す。そしてPallet_only_updateが1に設定され、Pallet_idが3に設定されている。これは、DS3の読み込み時において、Pallet_id=3で指示されるPDSを用いて、グラフィクス表示の更新を行う旨を示す。DS2～DS3に属するPCSは、以上のように

設定されているので、DS0に属するグラフィクスオブジェクトは、それぞれ別々のパレットデータを用いて、更新されてゆくことになる。

図２７は、これら4つのDisplay Setが読み込ませることで実現される、表示効果を現した図である。DS0に属するグラフィクスオブジェクトが再生装置に投入された段階で、字幕「忘れない」が表示され、それを構成する文字のうち、文字「忘」が赤色になる。次にDS1を投入した段階において、字幕「忘れない」を構成する文字のうち、文字「忘」,「れ」が赤色に、DS2を投入した段階で、文字「忘」,「れ」,「な」が赤色になる。そしてDS3を投入した段階で、全ての文字が赤色になる。Display Setが投入される毎に、字幕を構成する文字は、左に位置するものから順に、白色から赤色に変わってゆくので、音声の進行と共に、文字の色彩を代えてゆく表示効果を実現することができる。

これまでに説明したDisplay Set(PCS,WDS,PDS,ODS)のデータ構造は、プログラミング言語で記述されたクラス構造体のインスタンスであり、オーサリングを行う制作者は、Blu-ray Disc Read Only Formatに規定された構文に従い、クラス構造体を記述することで、BD-ROM上のこれらのデータ構造を得ることができる。　以上が本発明に係る記録媒体の実施形態である。続いて本発明に係る再生装置の実施形態について説明する。図２８は、本発明に係る再生装置の内部構成を示す図である。本発明に係る再生装置は、本図に示す内部に基づき、工業的に生産される。本発明に係る再生装置は、主としてシステムLSIと、ドライブ装置、マイコンシステムという3つのパーツからなり、これらのパーツを装置のキャビネット及び基板に実装することで工業的に生産することができる。システムLSIは、再生装置の機能を果たす様々な処理部を集積した集積回路である。こうして生産される再生装置は、BDドライブ１、Read Buffer２、PIDフィルタ３、Transport Buffer４a,b,c、周辺回路４d、ビデオデコーダ５、ビデオプレーン６、オーディオデコーダ７、グラフィクスプレーン８、CLUT部９、加算器１０、グラフィクスデコーダ１２、Coded Data Buffer１３、周辺回路１３a、Stream Graphics Processor１４、Object Buffer１５、Composition

Buffer１６、Graphical Controller１７から構成される。

BD-ROMドライブ１は、BD-ROMのローディング／リード／イジェクトを行い、BD-ROMに対するアクセスを実行する。

Read Buffer２は、FIFOメモリであり、BD-ROMから読み出されたTSパケットが先入れ先出し式に格納される。

PIDフィルタ３は、Read Buffer２から出力される複数TSパケットに対してフィルタリングを施す。PIDフィルタ３によるフィルタリングは、TSパケットのうち、所望のPIDをもつもののみをTransport Buffer４a,b,cに書き込むことでなされる。PIDフィルタ３によるフィルタリングでは、バッファリングは必要ではない。従って、PIDフィルタ３に入力されたTSパケットは、時間遅延なく、Transport Buffer４a,b,cに書き込まれる。

Transport Buffer４a,b,cは、PIDフィルタ３から出力されたTSパケットを先入れ先出し式に格納しておくメモリである。このTransport Buffer４aからTSパケットが取り出される速度を速度Rxとする

周辺回路４dは、Transport Buffer４a,b,cから読み出されたTSパケットを、機能セグメントに変換する処理を行うワイアロジックである。変換により得られた機能セグメントはCoded Data Buffer１３に格納される。

ビデオデコーダ５は、PIDフィルタ３から出力された複数TSパケットを復号して非圧縮形式のピクチャを得てビデオプレーン６に書き込む。

ビデオプレーン６は、動画用のプレーンメモリである。

オーディオデコーダ７は、PIDフィルタ３から出力されたTSパケットを復号して、非圧縮形式のオーディオデータを出力する。

グラフィクスプレーン８は、一画面分の領域をもったプレーンメモリであり、一画面分の非圧縮グラフィクスを格納することができる。

CLUT部９は、グラフィクスプレーン８に格納された非圧縮グラフィクスにおけるインデックスカラーを、PDSに示されるY,Cr,Cb値に基づき変換する。

加算器１０は、CLUT部９により色変換された非圧縮グラフィクスに、PDSに示されるT値(透過率)を乗じて、ビデオプレーン６に格納された非圧縮状態のピクチャデータと画素毎に加算し、合成画像を得て出力する。

グラフィクスデコーダ１２は、グラフィクスストリームをデコードして、非圧縮グラフィクスを得て、これをグラフィクスオブジェクトとしてグラフィクスプレーン８に書き込む。グラフィクスストリームのデコードにより、字幕やメニューが画面上に現れることになる。

グラフィクスデコーダ１２によるパイプラインは、DSnに属するグラフィクスオブジェクトをObject Buffer１５に書き込む処理、DSn+1に属するグラフィクスオブジェクトをObject Buffer１５から読み出す処理を同時に実行することでパイプラインは実行される。

このグラフィクスデコーダ１２は、Coded Data Buffer１３、周辺回路１３a、Stream Graphics Processor１４、Object Buffer１５、Composition Buffer１６、Graphical Controller１７から構成される。

Coded Data Buffer１３は、機能セグメントがDTS、PTSと共に格納されるバッファである。かかる機能セグメントは、Transport Buffer４a,b,cに格納されたトランスポートストリームの各TSパケットから、TSパケットヘッダ、PESパケットヘッダを取り除き、ペイロードをシーケンシャルに配列することにより得られたものである。取り除かれたTSパケットヘッダ、PESパケットヘッダのうち、PTS／DTSは、PESパケットと対応付けて格納される。

周辺回路１３aは、Coded Data Buffer１３－Stream Graphics Processor１４間の転送、Coded Data Buffer１３－Composition Buffer１６間の転送を実現するワイヤロジックである。この転送処理において現在時点がODSのDTSに示される時刻になれば、ODSを、Coded Data Buffer１３からStream Graphics Processor１４に転送する。また現在時刻がPCS、PDSのDTSに示される時刻になれば、PCS、PDSをComposition Buffer１６に転送するという処理を行う。

Stream Graphics Processor１４は、ODSをデコードして、デコード

により得られたインデックスカラーからなる非圧縮状態の非圧縮グラフィクスをグラフィクスオブジェクトとしてObject Buffer１５に書き込む。Stream Graphicsプロセッサ１４によるデコードは瞬時に行われ、デコードによりグラフィクスオブジェクトをStream Graphicsプロセッサ１４は一時的に保持する。Stream Graphicsプロセッサ１４によるデコードは瞬時になされるが、Stream Graphicsプロセッサ１４からObject Buffer１５への書き込みは、瞬時には終わらない。BD-ROM規格のプレーヤモデルでは、Object Buffer１５への書き込みは、128Mbpsという転送レートでなされるからである。Object Buffer１５への書き込み完了時点は、ENDセグメントのPTSに示されているので、このENDセグメントのPTSに示される時点が経過するまで、次のDSに対する処理を待つことになる。各ODSをデコードすることにより得られたグラフィクスオブジェクトの書き込みは、そのODSに関連付けられたDTSの時刻に開始し、ODSに関連付けられたPTSに示されるデコード終了時刻までに終了する。

書き込み時にあたってDSn側のグラフィックスデータと、DSn+1側のグラフィックスデータとに割り当てられているobject_idが別々の場合、Stream Graphicsプロセッサ１４はDSn側のグラフィックスデータと、DSn+1側のグラフィックスデータとを、Object Buffer１５における別々の領域に書き込む。これによりDSnのPCSにより参照されるグラフィクスオブジェクトは、DSn+1に属するグラフィクスオブジェクトにより上書きされることなく、パイプラインで表示に供される。両グラフィクスオブジェクトに割り当てられているobject_idが同じである場合、前記Stream Graphicsプロセッサ１４は、Object Buffer１５において先行DS側のグラフィックスデータが格納されている領域と同じ領域に、後続DS側のグラフィックスデータを上書きする。かかる場合、パイプラインは行わない。またDSに属するグラフィクスオブジェクトには、同一DSのPCSにより参照されているものと、参照されていないものとがある。Stream Graphicsプロセッサ１４は、PCSにより参照されているグラフィクスオブジェクトだけでなく、参照されてな

いグラフィクスオブジェクトを逐次デコードして、デコードにより得られたグラフィクスをObject Buffer１５に格納する。

Object Buffer１５は、ETSI EN 300 743標準規格におけるピクセルバッファに相当するバッファであり、Stream Graphics Processor１４のデコードにより得られたグラフィクスオブジェクトが配置される。Object Buffer１５は、グラフィクスプレーン８の2倍／4倍の大きさに設定せねばならない。何故ならScrollingを実現する場合を考えると、グラフィクスプレーン８の2倍、4倍のグラフィクスオブジェクトを格納しておかねばならないからである。

Composition Buffer１６は、PCS、PDSが配置されるメモリである。――処理すべきDisplay Setが2つあり、これらの有効区間が重複している場合、Compositionバッファ１６には処理すべきPCSが複数格納される。

Graphical Controller１７は、Graphicalコントローラ１７はPCSの解読を行い、PCSの解読結果に従って、グラフィクスオブジェクトのObject Buffer１５への書き込み、及び、Object Buffer１５からのグラフィクスオブジェクトの読み出し、グラフィクスオブジェクトの表示を実行する。Graphicalコントローラ１７による表示は、PCSを格納したPESパケットのPTSに示される時点において実行される。

オブジェクトCによる画面構成を実行するにあたって、Graphicsコントローラ１７は以下の処理を行う。Graphicsコントローラ１７は、オブジェクトCに記述されたグラフィクスオブジェクト識別子の参照値(object_id_ref)を読み取り、そのobject_id_refにより指示されるグラフィクスオブジェクトのうち、

object_cropping_horizontal_position,object_cropping_vertical_positionから、object_cropping_width、object_cropping_heightに示されるクロップ範囲を切り出して、グラフィックスプレーンのobject_horizontal_position,object_vertical_positionにより示される位置に書き込む。この参照値が、オブジェクトCと同じDisplay Setに属するグラフィクスオブジェクトを表している場合、同じDisplay

Setに属するODSがデコードされた後でないと、上述したグラフィックスプレーンへの書き込みは行えない。

一方、オブジェクトCが属するDisplay SetのComposition_stateがNormal Caseである場合、デコードにより得られたグラフィクスオブジェクトが、前もってオブジェクトバッファ内に格納されているので、ODSのデコードを待つことなく、上述したグラフィックスプレーンへの書き込みを行うことができる。

PalletOnly Displey Update時におけるGraphicsコントローラ１７の読み出し処理は以下の通りである。Display Setが、BD-ROMから読み出された場合、Graphicsコントローラ１７は、PCSのPallette_update_flagが、”1”であるか否かを判定する。もし1であるなら、PCSのpallet_idに記述されているパレットデータを用いた画素値変換を、CLUT部９に命じた上で、Object Buffer１５からグラフィクスオブジェクトを読み出し、グラフィクスプレーン８に格納させてゆく。CLUT部９に対するパレットデータの設定のみを行うことで、グラフィクスの更新を行うのである。

続いて、PIDフィルタ３、Transport Buffer４a,b,c、グラフィクスプレーン８、CLUT部９、Coded Data Buffer１３～Graphical Controller１７を構成するための、転送レート、バッファサイズの推奨値について説明する。図２９は、書込レートRx,Rc,Rd、グラフィクスプレーン８、Coded Data Buffer１３、Object Buffer１５、Composition Buffer１６のサイズを示す図である。

Object Buffer１５－グラフィクスプレーン８間の転送レートRcは、本装置において最も高い転送レートであり、ウィンドゥサイズ、フレームレートから256Mbps(=500Kバイト×29.97×2)と算出される。

Stream Graphics Processor１４－Object Buffer１５間の転送レートRd(Pixel Decoding Rate)は、Rcとは異なり、ビデオフレームの周期によるアップデートは要求されずRcの1/2,1/4でよい。故に128Mbps,64Mbpsになる。

Transport Buffer４a,b,c－Coded Data Buffer１３間のTransport

Buffer Leakレート Rxは、圧縮状態たるODSの転送レートである。従ってTransport Buffer Leakレートは、Rdに圧縮率を乗じた転送レートでよい。ODSの圧縮率を25%と仮定すれば、16Mbps(=64Mbps×25%)で足りる。

この図に示す転送レート、バッファ規模はあくまでもミニマムスタンダードであり、これより大きい値での実装を否定している訳ではない。

以上のように構成された再生装置において、各構成要素はパイプライン式にデコード処理を行うことができる。

図３０は、再生装置によるパイプライン処理を示すタイミングチャートである。第５段目は、BD-ROMにおけるDisplay Setを示し、第４段目は、Coded Data Buffer１３へのPCS、WDS、PDS、ODSの読出期間を示す。第３段目は、Stream Graphics Processor１４による各ODSのデコード期間を、第２段目はComposition Buffer１６の格納内容を、第１段目はGraphical Controller１７の処理内容を示す。

ODS1,2に付与されたDTS(デコード開始時刻)は、図中のt31,t32の時点を示している。デコード開始時刻がDTSに規定されているので、各ODSは、自身のDTSに示される時刻までにCoded Data Buffer１３に読み出されなければならない。そのためODS1の読み出しは、Coded Data Buffer１３へのODS1のデコード期間dp1の直前までに完了している。Coded Data Buffer１３へのODS2の読み出しは、ODS2のデコード期間dp2の直前までに完了している。

一方、ODS1,2に付与されたPTS(デコード終了時刻)は、図中のt32,t33の時点を示している。Stream Graphics Processor１４によるODS1のデコードはt32までに完了し、ODS2のデコードは、t33に示される時刻までに完了する。以上のように、Stream Graphics Processor１４は、各ODSのDTSに示される時刻までに、ODSをCoded Data Buffer１３に読み出し、Coded Data Buffer１３に読み出されたODSを、各ODSのPTSに示される時刻までに、デコードしてObject Buffer１５に書き込む。

本図の第１段目における期間cd1は、Graphics Controller１７がグラフィクスプレーン８をクリアするのに要する期間である。また期間td1は、Object Buffer１５上にえられたグラフィクスオブジェクトを、グラフィクスプレーン８に書き込むのに要する期間である。WDSのPTSは、この書き込みの開始にあたってのデッドラインを示し、PCSのPTSはこの書き込みの終了時点及び表示タイミングを示す。このPCSのPTSに示される時点になれば、対話画面を構成する非圧縮グラフィクスがグラフィクスプレーン８上に得られることになる。この非圧縮グラフィクスの色変換をCLUT部９に行わせ、ビデオプレーン６に格納されている非圧縮ピクチャとの合成を加算器１０に行わせれば、合成画像が得られることになる。

グラフィクスデコーダ１２において、Graphics Controller１７がグラフィクスプレーン８のクリアを実行している間においても、Stream Graphics Processor１４のデコードは継続して行われる。以上のようなパイプライン処理により、グラフィクスの表示を迅速に実施することができる。

図３０では、グラフィックスプレーンのクリアが、ODSのデコードより早く終わる場合を想定したが、図３１は、ODSのデコードが、グラフィックスプレーンのクリアより早く終わる場合を想定したパイプライン処理を示すタイミングチャートである。この場合、ODSのデコードが完了した段階では、グラフィックスプレーンへの書き込みを実行することができず、グラフィックスプレーンのクリアが完了した時点で、デコードにより得られたグラフィクスをグラフィックスプレーンに書き込むことができる。

再生装置におけるバッファ占有量の時間的遷移について図３２を参照しながら説明する。図３２は、図２８におけるCompositionバッファ１６、Object Buffer１５、Coded Dataバッファ１３、グラフィクスプレーン８の時間的遷移を示すタイミングチャートである。本図は、第１段目から第４段目までに、グラフィクスプレーン８、Object Buffer１５、Coded Dataバッファ１３、Compositionバッファ１６に

おける占有量の時間的遷移を示している。この占有量の時間的遷移は、横軸を時間軸とし、縦軸を占有量とした折れ線グラフの表記で表現している。

図３２の第４段目は、Compositionバッファ１６における占有量の時間的遷移を示す。本図に示すようにCompositionバッファ１６の時間的遷移は、Coded Dataバッファ１３から出力されPCSが格納されることによる単調増加vf0を含む。

第３段目は、Coded Dataバッファ１３における占有量の時間的遷移を示す。本図に示すようにCoded Dataバッファ１３の時間的遷移は、ODSが格納されることによる単調増加Vf1,Vf2と、格納されたODSが順次Stream Graphicsプロセッサ１４により取り出されることによる単調減少Vg1,Vg2とを含む。単調増加Vf1,Vf2の傾きは、Transportバッファ４a,b,cからCoded Dataバッファ１３への出力レートRxに基づき、単調減少Vg1,Vg2の傾きは、Stream Graphicsプロセッサ１４によるデコードであり、瞬時に実行される。つまりODSに対するデコードは瞬時に行われ、Stream Graphicsプロセッサ１４は、デコードにより得られた非圧縮グラフィクスを保持する。Stream Graphicsプロセッサ１４からObject Buffer１５への伝送路の書込レートは128Mbpsであるため、この書込レートにより、Object Buffer１５の占有量は増加する。

第２段目は、Object Buffer１５における占有量の時間的遷移を示す。本図に示すようにObject Buffer１５の時間的遷移は、Stream Graphicsプロセッサ１４から出力されたODSが格納されることによる単調増加Vh1,Vh2を含む。単調増加Vh1,Vh2の傾きは、Stream Graphicsプロセッサ１４からObject Buffer１５への転送レートRdに基づく。第３段目の単調減少が生じる期間及びの第２段目の単調増加が生ずる期間が、デコード期間である。このデコード期間の始期は、ODSのDTSに示されており、デコード期間の終期は、ODSのPTSに示されている。このODSのPTSに示される期間までに、非圧縮のグラフィクスがobject buffer１５に格納されれば、ODSに対するデコードは完了した

ことになる。ODSのPTSに示される期間までに、非圧縮のグラフィクスがobject buffer１５に格納されることが、必須の要件であり、このデコード期間における単調増加、単調減少はどのようなものであってもよい。

第１段目は、グラフィクスプレーン８における占有量の時間的遷移を示す。本図に示すようにグラフィクスプレーン８の時間的遷移は、Object Buffer１５から出力されたデコード済みODSが格納されることによる単調増加Vf3を含む。単調増加Vf3の傾きは、Object Buffer１５からグラフィクスプレーン８への転送レートRcに基づく。この単調増加の終期は、PCSのPTSに示されている。

ODSに付与されたDTS、PTS、ICSに付与されたDTS、PTS、そして図２９に示した各バッファのサイズ、転送レートを用いれば、本図のようなグラフを作図することにより、BD-ROMにて供給すべきAVClipの再生時において、各バッファの状態がどのように変化するかが、オーサリングの段階で明らかになる。

このバッファ状態の遷移は、DTS、PTSに示される値を書き換えることで、調整することが可能なので、再生装置側のデコーダのスペックを越えるような復号負荷の発生を回避したり、再生にあたってのバッファオーバーフローの回避することができる。そのため再生装置の開発にあたってのハードウェア、ソフトウェアの実装が簡易になる。

以上が再生装置の内部構成である。続いてグラフィクスデコーダ１２を、どのようにして実装するかについて説明する。グラフィクスデコーダ１２は、図３３の処理手順を行うプログラムを作成し、CPUに実行させることにより実装可能である。以降、図３３を参照しながら、制御部２０の処理手順について説明する。

図３３は、機能セグメントのロード処理の処理手順を示すフローチャートである。本フローチャートにおいてSegmentKとは、AVClipの再生時において、読み出されたSegment(PCS,WDS,PDS,ODS)のそれぞれを意味する変数であり、無視フラグは、このSegmentKを無視するかロードするかを切り換えるフラグである。本フローチャートは、無視フラ

グを0に初期化した上で、ステップＳ２１～Ｓ２４、ステップＳ２７～Ｓ３１の処理を全てのSegmentKについて繰り返すループ構造を有している(ステップＳ２５、ステップＳ２６)。

ステップＳ２１は、SegmentKがPCSであるか否かの判定であり、もしSegmentKがPCSであれば、ステップＳ２７、ステップＳ２８の判定を行う。

ステップＳ２２は、無視フラグが1かどうかの判定である。無視フラグが0であるならステップＳ２３に移行し、1であるならステップＳ２４に移行する。無視フラグが1であれば(ステップＳ２２でNo)、ステップＳ２３においてSegmentKをCoded Data Buffer１３にロードする。

無視フラグが0に設定されていれば(ステップＳ２２がNo)、ステップＳ２４においてSegmentKが無視される。これによりDSに属する機能セグメントは全て、ステップＳ２２がNoになって、無視されることになる(ステップＳ２４)。

このように、SegmentKが無視されるか、ロードされるかは、無視フラグの設定により決まる。ステップＳ２７～Ｓ３１、Ｓ３４、Ｓ３５は、この無視フラグを設定する処理である。

ステップＳ２７は、PCSにおけるcomposition_stateがAcquisition Pointであるか否かの判定である。SegmentKがAcquisition Pointであるなら、ステップＳ２８に移行し、SegmentKがもしEpoch StartかNormal Caseであるなら、ステップＳ３１に移行する。

ステップＳ２８は、先行するDSがグラフィクスデコーダ１２内のどれかのバッファ(Coded Data Buffer１３、Stream Graphicsプロセッサ１４、Object Buffer１５、Composition Buffer１６)に存在するかどうかの判定であり、ステップS.２７がYesである場合に実行される。グラフィクスデコーダ１２内にDSが存在しないケースとは、頭出しがなされたケースをいう。この場合、Acquisition PointたるDSから、表示を開始せねばならないので、ステップＳ３０に移行する(ステップＳ２８でNo)。

ステップＳ３０は、無視フラグを0に設定し、ステップＳ２２に移行する。

グラフィクスデコーダ１２内にDSが存在するケースとは、通常再生がなされたケースをいう。この場合、ステップＳ２９に移行する(ステップＳ２８でYes)。ステップＳ２９は、無視フラグを1に設定し、ステップＳ２２に移行する。

ステップＳ３１は、PCSにおけるcomposition_stateがNormal Caseであるか否かの判定である。もしNormal Caseであるなら、ステップＳ３４に移行する。SegmentKがEpoch Startであるなら、ステップＳ３０において無視フラグを0に設定する。 ステップＳ３４は、ステップＳ２８と同じであり、先行するDSがグラフィクスデコーダ１２内に存在するかどうかの判定を行う。もし存在するなら、無視フラグを0に設定する(ステップＳ３０)。存在しないなら、元々、対話画面を構成する充分な機能セグメントが得られないため、無視フラグを1に設定する(ステップＳ３５)。かかるフラグ設定により、先行するDSがグラフィクスデコーダ１２に存在しない場合、Normal Caseを構成する機能セグメントは無視されることになる。

DSが、図３４のように多重化されている場合を想定して、DSの読み出しがどのように行われるかを説明する。図３４の一例では、3つのDSが動画と多重化されている。この3つのDSのうち、初めのDS1は、composition_stateがEpoch_Start、DS10はAcquisition Point、DS20は、Normal Caseである。

かかる3つのDSが、動画と多重化されているAVClipにおいて、ピクチャデータpt10からの頭出しが矢印am1に示すように行われたものとする。この場合、頭出し位置に最も近いDS10が、図３３のフローチャートの対象となる。ステップＳ２７においてcomposition_stateはAcquisition Pointと判定されるが、先行するDSはCoded Data Buffer１３上に存在しないため、無視フラグは0に設定され、このDS10が図３５の矢印md1に示すように再生装置のCoded Data Buffer１３にロードされる。一方、頭出し位置がDS10の存在位置より後である場合(図

３４の矢印am2)、DS20は、Normal CaseのDisplay Setであり、先行するDS20はCoded Data Buffer１３に存在しないので、このDisplay Setは、無視されることになる(図３５の矢印md2)。

図３６のように通常再生が行われた場合のDS1,10,20のロードは、図３６に示すものとなる。3つのDSのうち、PCSのcomposition_stateがEpoch StartであるDS1は、そのままCoded Data Buffer１３にロードされるが(ステップＳ２３)、PCSのcomposition_stateがAcquisition PointであるDS10については、無視フラグが1に設定されるため(ステップＳ２９)、これを構成する機能セグメントはCoded Data Buffer１３にロードされず無視される(図３７の矢印rd2,ステップＳ２４)。またDS20については、PCSのcomposition_stateはNormal Caseなので、Coded Data Buffer１３にロードされる(図３７の矢印rd3)。

続いてGraphical Controller１７の処理手順について説明する。図３８～図４０は、Graphical Controller１７の処理手順を示すフローチャートである。

ステップＳ４１～ステップＳ４４は、本フローチャートのメインルーチンであり、ステップＳ４１～ステップＳ４４に規定した何れかの事象の成立を待つ。

ステップＳ４１は、現在の再生時点がPCSのDTS時刻になっているか否かの判定であり、もしなっていれば、ステップＳ４５～ステップＳ５３の処理を行う。

ステップＳ４５は、PCSのcomposition_stateが、Epoch_Startであるか否かの判定であり、もしEpoch Startであるなら、ステップＳ４６においてグラフィクスプレーン８を全クリアする。それ以外であるなら、ステップＳ４７においてWDSのwindow_horizontal_position、window_vertival_position、window_width、window_heightに示されるwindowをクリアする。

ステップＳ４８は、ステップＳ４６又はステップＳ４７でのクリア後の実行されるステップであり、任意のODSxのPTS時刻が既に経過し

ているか否かの判定である。つまり、グラフィクスプレーン８全体のクリアにあたっては、そのクリア時間に長時間を費するので、あるODS(ODSx)のデコードが既に完了していることもある。ステップＳ４８はその可能性を検証している。もし経過していないなら、メインルーチンにリターンする。どれかのODSのデコード時刻を経過しているなら、ステップＳ４９～ステップＳ５１を実行する。ステップＳ４９は、object_cropped_flagが0を示しているか否かの判定であり、もし0を示しているなら、グラフィクスオブジェクトを非表示とする(ステップＳ５０)。

もし0を示していないなら、object_cropping_horizontal_position、object_cropping_vertival_position、cropping_width、cropping_heightに基づきクロップされたグラフィクスオブジェクトを、グラフィクスプレーン８のwindowにおいてobject_horizontal_position,object_vertival_positionに示される位置に書き込む(ステップＳ５１)。以上の処理により、ウィンドゥに1つ以上のグラフィクスオブジェクトが描かれることになる。

ステップＳ５２は、別のODSyのPTS時刻が経過しているか否かの判定である。ODSxをグラフィクスプレーン８に書き込んでいる際、別のODSのデコードが既に完了していれば、このODSyをODSxにして(ステップＳ５３)、ステップＳ４９に移行する。これにより、別のODSに対しても、ステップＳ４９～Ｓ５１の処理が繰り返し行われる。

次に図３９を参照して、ステップＳ４２、ステップＳ５４～ステップＳ５９について説明する。

ステップＳ４２は、現在の再生時点がWDSのPTSであるか否かの判定であり、もしWDSのPTSであるなら、ステップＳ５４においてウィンドゥが1つであるか否かを判定し、もし2つであれば、メインルーチンにリターンする。ウィンドゥが1つであるなら、ステップＳ５５～ステップＳ５９のループ処理を行う。このループ処理は、ウィンドゥに表示される2つのグラフィクスオブジェクトのそれぞれについて、ステップＳ５７～ステップＳ５９を実行するというものである。ステップ

Ｓ５７は、object_cropped_flagが0を示しているか否かの判定であり、もし0を示しているなら、グラフィクスオブジェクトを非表示とする（ステップＳ５８）。

もし0を示していないなら、object_cropping_horizontal_position、object_cropping_vertival_position、cropping_width、cropping_heightに基づきクロップされたグラフィクスオブジェクトを、グラフィクスプレーン８のwindowにおいてobject_horizontal_position,object_vertival_positionに示される位置に書き込む（ステップＳ５９）。以上の処理を繰り返せば、ウィンドゥに1つ以上のグラフィクスオブジェクトが描かれることになる。

ステップＳ４４は、現在の再生時点がPCSのPTSに示される時点であるかの判定であり、もしそうであるなら、ステップＳ６０においてPallet_update_flagが1を示しているか否かを判定する。もし1を示しているなら、pallet_idに示されるPDSをCLUT部に設定する（ステップＳ６１）。0を示しているなら、ステップＳ６１をスキップする。

その後、グラフィクスプレーン８におけるグラフィクスオブジェクトの色変換をCLUT部に行わせて、動画像と合成する（ステップＳ６２）。

次に図４０を参照して、ステップＳ４３、ステップＳ６４～ステップＳ６６について説明する。

ステップＳ４３は、現在の再生時点がODSのPTSであるか否かの判定であり、もしODSのPTSであるなら、ステップＳ６３においてウィンドゥが2つであるか否かを判定する。もし1つであれば、メインルーチンにリターンする。

ステップＳ４３及びステップＳ６３の判定は以下の意味をもつ。つまりウィンドゥが2つある場合、それぞれのウィンドゥには、1つずつグラフィクスオブジェクトが表示される。そうすると、それぞれのODSのデコードが完了する度に、デコードにより得られたグラフィクスオブジェクトを、グラフィックスプレーンに書き込んでゆく処理が必要になる（例：図１９（ｂ）のケース）。そこで現在時点がODSのPTSに示される時点であり、ウィンドゥが2つであるなら、個々のグラフィク

スオブジェクトをグラフィックスプレーンに書き込むべく、ステップＳ６４～ステップＳ６６を実行する。ステップＳ６４は、object_cropped_flagが0を示しているか否かの判定であり、もし示しているなら、グラフィクスオブジェクトを非表示とする(ステップＳ６５)。

もし0を示していないなら、object_cropping_horizontal_position、object_cropping_vertival_position、cropping_width、cropping_heightに基づきクロップされたグラフィクスオブジェクトを、グラフィクスプレーン８のwindowにおいてobject_horizontal_position,object_vertival_positionに示される位置に書き込む(ステップＳ６６)。以上の処理を繰り返せば、各ウィンドゥにグラフィクスオブジェクトが描かれることになる。

(第２実施形態)

本実施形態は、BD-ROMの製造工程に関する実施形態である。図４１は、第１実施形態に示したPCSを作成するための製造工程を示す図である。

BD-ROMの制作工程は、動画収録、音声収録等の素材作成を行う素材制作工程Ｓ２０１、オーサリング装置を用いて、アプリケーションフォーマットを生成するオーサリング工程Ｓ２０２、BD-ROMの原盤を作成し、プレス・貼り合わせを行って、BD-ROMを完成させるプレス工程Ｓ２０３を含む。

これらの工程のうち、BD-ROMを対象としたオーサリング工程は、以下のステップＳ２０４～ステップＳ２１０を含む。

ステップＳ２０４において制御情報、ウィンドゥ定義情報、パレット定義情報、グラフィクスを記述し、ステップＳ２０５では、制御情報、ウィンドゥ定義情報、パレット定義情報、グラフィクスを機能セグメントに変換する。そしてステップＳ２０６において同期したいピクチャが出現するタイミングに基づき、PCSのPTSを設定し、ステップＳ２０７では、PTS[PCS]の値に基づき、DTS[ODS],PTS[ODS]を設定する。ステップＳ２０８において、DTS[ODS]の値に基づき、

DTS[PCS],PTS[PDS],DTS[WDS],PTS[WDS]を設定し、ステップＳ２０９では、プレーヤモデルにおける各バッファの占有量の時間的遷移をグラフ化する。ステップＳ２１０では、グラフ化された時間的遷移がプレーヤモデルの制約を満たすか否かを判定し、もし満たさないなら、ステップＳ２１１において各機能セグメントのDTS、PTSを書き換える。もし満たすならステップＳ２１２においてグラフィクスストリームを生成し、ステップＳ２１３においてグラフィクスストリームを別途生成されたビデオストリーム、オーディオストリームと多重してAVClipを得る。以降、AVClipをBD-ROMのフォーマットに適合させることにより、アプリケーションフォーマットが完成する。

（備考）

以上の説明は、本発明の全ての実施行為の形態を示している訳ではない。下記(A)(B)(C)(D)・・・・・の変更を施した実施行為の形態によっても、本発明の実施は可能となる。本願の請求項に係る各発明は、以上に記載した複数の実施形態及びそれらの変形形態を拡張した記載、ないし、一般化した記載としている。拡張ないし一般化の程度は、本発明の技術分野の、出願当時の技術水準の特性に基づく。

(A)全ての実施形態では、本発明に係る記録媒体をBD-ROMとして実施したが、本発明の記録媒体は、記録されるグラフィクスストリームに特徴があり、この特徴は、BD-ROMの物理的性質に依存するものではない。グラフィクスストリームを記録しうる記録媒体なら、どのような記録媒体であってもよい。例えば、

DVD-ROM,DVD-RAM,DVD-RW,DVD-R,DVD+RW,DVD+R,CD-R,CD-RW等の光ディスク、PD,MO等の光磁気ディスクであってもよい。また、コンパクトフラッシュカード、スマートメディア、メモリスティック、マルチメディアカード、PCM-CIAカード等の半導体メモリカードであってもよい。フレシキブルディスク、SuperDisk,Zip,Clik!等の磁気記録ディスク(i)、ORB,Jaz,SparQ,SyJet,EZFley,マイクロドライブ等のリムーバルハードディスクドライブ(ii)であってもよい。更に、機器内蔵型のハードディスクであってもよい。

(B)全ての実施形態における再生装置は、BD-ROMに記録されたAVClipをデコードした上でTVに出力していたが、再生装置をBD-ROMドライブのみとし、これ以外の構成要素をTVに具備させてもい、この場合、再生装置と、TVとをIEEE1394で接続されたホームネットワークに組み入れることができる。また、実施形態における再生装置は、テレビと接続して利用されるタイプであったが、ディスプレィと一体型となった再生装置であってもよい。更に、各実施形態の再生装置において、処理の本質的部分をなすシステムLSI(集積回路)のみを、実施としてもよい。これらの再生装置及び集積回路は、何れも本願明細書に記載された発明であるから、これらの何れの態様であろうとも、第1実施形態に示した再生装置の内部構成を元に、再生装置を製造する行為は、本願の明細書に記載された発明の実施行為になる。第1実施形態に示した再生装置の有償・無償による譲渡(有償の場合は販売、無償の場合は贈与になる)、貸与、輸入する行為も、本発明の実施行為である。店頭展示、カタログ勧誘、パンフレット配布により、これらの譲渡や貸渡を、一般ユーザに申し出る行為も本再生装置の実施行為である。

(C)各フローチャートに示したプログラムによる情報処理は、ハードウェア資源を用いて具体的に実現されていることから、上記フローチャートに処理手順を示したプログラムは、単体で発明として成立する。全ての実施形態は、再生装置に組み込まれた態様で、本発明に係るプログラムの実施行為についての実施形態を示したが、再生装置から分離して、第1実施形態に示したプログラム単体を実施してもよい。プログラム単体の実施行為には、これらのプログラムを生産する行為(1)や、有償・無償によりプログラムを譲渡する行為(2)、貸与する行為(3)、輸入する行為(4)、双方向の電子通信回線を介して公衆に提供する行為(5)、店頭、カタログ勧誘、パンフレット配布により、プログラムの譲渡や貸渡を、一般ユーザに申し出る行為(6)がある。

(D)各フローチャートにおいて時系列に実行される各ステップの「時」の要素を、発明を特定するための必須の事項と考える。そうする

と、これらのフローチャートによる処理手順は、再生方法の使用形態を開示していることがわかる。各ステップの処理を、時系列に行うことで、本発明の本来の目的を達成し、作用及び効果を奏するよう、これらのフローチャートの処理を行うのであれば、本発明に係る記録方法の実施行為に該当することはいうまでもない。

(E)BD-ROMに記録するにあたって、AVClipを構成する各TSパケットには、拡張ヘッダを付与しておくことが望ましい。拡張ヘッダは、TP_extra_headerと呼ばれ、『Arribval_Time_Stamp』と、『copy_permission_indicator』とを含み4バイトのデータ長を有する。TP_extra_header付きTSパケット(以下EX付きTSパケットと略す)は、32個毎にグループ化されて、3つのセクタに書き込まれる。32個のEX付きTSパケットからなるグループは、6144バイト(=32×192)であり、これは3個のセクタサイズ6144バイト(=2048×3)と一致する。3個のセクタに収められた32個のEX付きTSパケットを"Aligned Unit"という。

IEEE1394を介して接続されたホームネットワークでの利用時において、再生装置は、以下のような送信処理にてAligned Unitの送信を行う。つまり送り手側の機器は、Aligned Unitに含まれる32個のEX付きTSパケットのそれぞれからTP_extra_headerを取り外し、TSパケット本体をDTCP規格に基づき暗号化して出力する。TSパケットの出力にあたっては、TSパケット間の随所に、isochronousパケットを挿入する。この挿入箇所は、TP_extra_headerのArribval_Time_Stampに示される時刻に基づいた位置である。TSパケットの出力に伴い、再生装置はDTCP_Descriptorを出力する。DTCP_Descriptorは、TP_extra_headerにおけるコピー許否設定を示す。ここで「コピー禁止」を示すようDTCP_Descriptorを記述しておけば、IEEE1394を介して接続されたホームネットワークでの利用時においてTSパケットは、他の機器に記録されることはない。

(F)各実施形態におけるデジタルストリームは、BD-ROM規格のAVClipであったが、DVD-Video規格、DVD-Video Recording規格の

VOB(Video Object)であってもよい。VOBは、ビデオストリーム、オーディオストリームを多重化することにより得られたISO/IEC13818-1規格準拠のプログラムストリームである。またAVClipにおけるビデオストリームは、MPEG4やWMV方式であってもよい。更にオーディオストリームは、Linear-PCM方式、Dolby-AC3方式、MP3方式、MPEG-AAC方式、dts方式であってもよい。

(G)各実施形態における映画作品は、アナログ放送で放送されたアナログ映像信号をエンコードすることにより得られたものでもよい。デジタル放送で放送されたトランスポートストリームから構成されるストリームデータであってもよい。

またビデオテープに記録されているアナログ／デジタルの映像信号をエンコードしてコンテンツを得ても良い。更にビデオカメラから直接取り込んだアナログ／デジタルの映像信号をエンコードしてコンテンツを得ても良い。他にも、配信サーバにより配信されるデジタル著作物でもよい。

(H)第１実施形態～第２実施形態に示したグラフィックスオブジェクトは、ランレングス符号化されたラスタデータである。グラフィックスオブジェクトの圧縮･符号化方式にランレングス符号方式を採用したのは、ランレングス符号化は字幕の圧縮･伸長に最も適しているためである。字幕には、同じ画素値の水平方向の連続長が比較的長くなるという特性があり、ランレングス符号化による圧縮を行えば、高い圧縮率を得ることができる。また伸長のための負荷も軽く、復号処理のソフトウェア化に向いているからである。しかし、グラフィックスオブジェクトにランレングス符号化方式を採用したというのは、本発明の必須事項ではなく、グラフィックスオブジェクトはPNGデータであってもよい。またラスタデータではなくベクタデータであってもよい、更に透明な絵柄であってもよい。

(I)PCSによる表示効果の対象は、装置側の言語設定に応じて選ばれた字幕のグラフィクスであってもよい。これにより、現状のDVDにおいて動画像本体で表現していたような文字を用いた表示効果を、装置

側の言語設定に応じて表示された字幕グラフィクスで実現することができるので、実用上の価値は大きい。

またPCSによる表示効果の対象は、装置側のディスプレイ設定に応じて選ばれた字幕グラフィクスであってもよい。つまり、ワイドビジョン、パンスキャン、レターボックス用といった様々な表示モード用のグラフィクスがBD-ROMに記録されており、装置側は自身に接続されたテレビの設定に応じてこれらの何れかを選んで表示する。この場合、そうして表示された字幕グラフィクスに対し、PCSに基づく表示効果をほどこすので、見栄えがよくなる。これにより、動画像本体で表現していたような文字を用いた表示効果を、装置側のディスプレイ設定に応じて表示された字幕で実現することができるので、実用上の価値は大きい。

(J)第1実施形態ではグラフィックスプレーンへの書込レートRcは、1ビデオフレーム内にグラフィックスプレーンクリア及び再描画が可能になるよう、windowのサイズを全体の25%に定めたが、これらクリア・再描画が垂直帰線期間に完遂するよう、Rcを定めても良い。垂直帰線期間は1/29.93秒の25%と仮定すると、Rcは1Gbpsになる。Rcをこのように設定することでグラフィクス表示はスムーズになされるので、実用上の効果は大きい。

また垂直帰線期間での書き込みに加え、ラインスキャンに同期した書き込みを併用してもよい。これにより、Rc=256Mbpsの書込レートであっても、スムーズな表示の実現が可能になる。

(K)各実施形態において再生装置には、グラフィックスプレーンを実装したが、このグラフィックスプレーンに代えて、一ライン分の非圧縮画素を格納するラインバッファを具備してもよい。映像信号への変換は水平行(ライン)毎に行われるので、このラインバッファさえ具備していれば、この映像信号への変換は行なえるからである。

(L)グラフィクスたる字幕は、映画の台詞を表す文字列であるとして説明を進めたが、動画と緻密に同期して表示されるものであればなんであってもよい。例えばイラストや絵柄、キャラクタ、シンボルマ

ークを描いたものであってもよい。商標を構成するような図形,文字,色彩の組合せや、国の紋章,旗章,記章,国家が採用する監督/証明用の公の記号・印章、政府間国際機関の紋章,旗章,記章,特定商品の原産地表示を含んでいてもよい。

(M)第１実施形態では、字幕を画面の上側、下側に横書きで表示するものとして、ウィンドゥをグラフィックスプレーンの上側、下側に定義したが、字幕を画面の右側、左側に表示するものとして、ウィンドゥをグラフィックスプレーンの右側、左側に定義してもよい。こうすることにより、日本語字幕を縦書きで表示することができる。

(O)グラフィクスデコーダ１２が、DSn及びDSn+1に対する処理をパイプライン式に行うのは、DSn及びDSn+1が、グラフィクスストリームにおける同じEpochに帰属している場合であり、前記DSn及びDSn+1が、互いに異なるEpochに属している場合、DSnにおけるグラフィクス表示を開始した後に、DSn+1に対する処理を開始する。

またグラフィクスストリームには、動画との同期を主たる目的としたプレゼンテーション系のものと、対話的な表示を主目的としたインタラクティブ系のものとがあり、前記グラフィクスデコーダは、グラフィクスストリームがプレゼンテーション系である場合に、2つのDSに対するパイプライン式を行い、グラフィクスストリームがインタラクティブ系である場合、2つのDSに対するパイプライン式を行わない。

(P)Display Set内にグラフィクスデータを配列させるにあたって、図２０では、同じDisplay Set内のPCSにより参照されるグラフィクスデータを、参照されないグラフィクスデータの前に配列したが、複数の参照グラフィクスデータ、及び、複数の非参照グラフィクスデータを、それぞれ、object_idにより指示される識別番号順に配列してもよい。また複数の参照グラフィクスデータは、スキャンライン順に配列してもよい。スキャンラインとは、ディスプレイでの表示時において、走査される順番を表す。通常ディスプレイにおける走査は、左上→右下方向に向けてなされるから、表示座標が左上にあるグラフィクスデータを前に、表示座標が右下にあるグラフィクスデータを後に配

置するのである。こうすることで、グラフィクスデータ表示の迅速化を図ることができる。

上述した変更実施は可能であるものの、請求項に係る各発明は、従来技術の技術的課題を解決するための手段を反映したものであるから、請求項に係る各発明の技術範囲は、従来技術の技術的課題解決が当業者により認識される技術範囲を超えることはない。故に、本願の請求項に係る各発明は、詳細説明の記載と、実質的な対応関係を有する。

産業上の利用可能性

本発明に係る記録媒体及び再生装置は、上記実施形態に内部構成が開示されており、この内部構成に基づき工業的に量産することが可能なので、資質において工業上用することができる。このことから本発明に係る記録媒体及び再生装置は、産業上利用可能性を有する。

請求の範囲

1．動画ストリームとグラフィクスストリームとを多重化することにより得られたデジタルストリームが記録されている記録媒体であって、

グラフィクスストリームは、パケットの配列であり、

パケットには、グラフィクスデータを格納したものと、制御情報を格納したものとがあり、

制御情報は、パケット列において自身より前方に存在するグラフィクスデータを、所定のタイミングで動画ストリームと合成して表示する旨を示す、ことを特徴とする記録媒体。

2．グラフィクスストリームにおけるパケットは、複数のディスプレィセットの何れかに分類されており、個々のディスプレィセットは、グラフィクス表示を実現する一個のデータの集合であり、

前記グラフィクスデータ及び制御情報は、

別々のディスプレィセットに属しており、

前記グラフィクスデータは、同じディスプレィセットに属する制御情報からは参照されていない、非参照グラフィクスデータである

ことを特徴とする請求項1記載の記録媒体。

3．制御情報は、完結した状態でパケットに格納されており、

前記所望のタイミングは、前記パケットのタイムスタンプに示されている、ことを特徴とする請求項1記載の記録媒体。

4．前記制御情報は、パレットデータと組みになって記録媒体に記録されており、

前記制御情報による表示は、パレットデータを用いたグラフィクスデータの色変換を、再生装置に命じる旨を示す、ことを特徴とする請求項1記載の記録媒体。

5．制御情報により示される表示は、グラフィクスデータの2回目以降の表示であり、

制御情報は、更新フラグを有し、

更新フラグは、色変換に用いるパレットデータのみを用いて表示更

新を行う旨を示す、ことを特徴とする請求項４記載の記録媒体。

６．前記グラフィクスストリームはウィンドゥ情報を含み、

ウィンドゥ情報は、グラフィクス表示のための描画領域を特定する情報であり、動画ストリームと合成する際の画面上でのウィンドゥの位置、縦幅、横幅を示し、

前記制御情報による表示位置は、

前記グラフィクス表示がウィンドゥに収まるように規定されている

ことを特徴とする請求項１記載の記録媒体。

７．ビデオストリーム、グラフィクスストリームが多重化されたデジタルストリームを再生する再生装置であって、

ビデオストリームをデコードして動画像を得るビデオデコーダと、

グラフィクスストリームをデコードしてグラフィクスを得て、動画像に合成するグラフィクスデコーダとを備え、

グラフィクスデコーダは、

記録媒体から新たな制御情報が記録媒体から読み込まれれば、その制御情報に基づき、前もって送り込まれたグラフィクスデータを表示する

ことを特徴とする再生装置。

８．グラフィクスストリームはグラフィクス表示を実現するデータの集合であるディスプレィセットを複数含み、

コントローラは、

1つのディスプレィセットが記録媒体から読み出されれば、そのディスプレィセットに属するグラフィクスデータであって、同じディスプレィセットに属する制御情報からは参照されていない、非参照グラフィクスデータをオブジェクトバッファに格納しておき、

前記前もって送り込まれたグラフィクスデータは、オブジェクトバッファに格納されている非参照グラフィクスデータである

ことを特徴とする請求項７記載の再生装置。

９．記録媒体から新たに読み出される制御情報は、完結した状態で

パケットに格納されており、

前記グラフィクスデコーダによる表示は、

ビデオストリームの再生が、パケットのタイムスタンプに示されている時点に到達した際に行われる

ことを特徴とする請求項７記載の再生装置。

１０．前記デコードにより得られた非圧縮グラフィクスは、コード値を用いて表現され、

前記再生装置は、

グラフィクスを表すコード値を、画素値に変換するルックアップテーブル部を備え、

制御情報は、パレットデータと組みになって、記録媒体から読み出され、

前記グラフィクスデコーダによる表示とは、

記録媒体から読み出されたパレットデータを、ルックアップテーブル部に設定し、そのパレットデータを用いた画素値変換を、ルックアップテーブル部に命じることである

ことを特徴とする請求項７記載の再生装置。

１１．グラフィクスデコーダによる表示は、既に表示されているグラフィクスの更新であり、

制御情報は、更新フラグを有し、

更新フラグが所定の値であるなら、ルックアップテーブル部に対するパレットデータの設定のみを行うことで、グラフィクスの更新を行う、ことを特徴とする請求項１０記載の再生装置。

１２．前記グラフィクスストリームはウィンドゥ情報を有し、

ウィンドゥ情報は、各ピクチャを表示するための画面の一部を、グラフィクス表示用のウィンドゥとして指定する情報であり、

グラフィクスデコーダによるグラフィクス描画は、

画面においてウィンドゥ情報に示されるウィンドゥをクリアする処理、及び、画面におけるウィンドゥに対しグラフィクスを書き込む処理を含むことでなされる、請求項７記載の再生装置。

１３．記録媒体の記録方法であって、

アプリケーションデータを作成するステップと、

作成したデータを記録媒体に記録するステップとを有し、

前記アプリケーションデータは、

動画ストリームとグラフィクスストリームとを多重化することにより得られたデジタルストリームを含み、

グラフィクスストリームは、パケットの配列であり、

パケットには、グラフィクスデータを格納したものと、制御情報を格納したものとがあり、

制御情報は、パケット列において自身より前方に存在するグラフィクスデータを、所定のタイミングで動画ストリームと合成して表示する旨を示す、

ことを特徴とする記録方法。

１４．ビデオストリーム、グラフィクスストリームが多重化されたデジタルストリームの再生をコンピュータに行わせるプログラムであって、

ビデオストリームをデコードして動画像を得るステップと、

グラフィクスストリームをデコードしてグラフィクスを得て、動画像に合成するステップとを備え、

グラフィクスを得るステップは、

記録媒体から新たな制御情報が記録媒体から読み込まれれば、その制御情報に基づき、前もって送り込まれたグラフィクスデータを表示する

処理をコンピュータに行わせることを特徴とするプログラム。

１５．ビデオストリーム、グラフィクスストリームが多重化されたデジタルストリームの再生をコンピュータに行わせる再生方法であって、

ビデオストリームをデコードして動画像を得るステップと、

グラフィクスストリームをデコードしてグラフィクスを得て、動画像に合成するステップとを備え、

グラフィクスを得るステップは、

記録媒体から新たな制御情報が記録媒体から読み込まれれば、その制御情報に基づき、前もって送り込まれたグラフィクスデータを表示する

ことを特徴とする再生方法。

図1

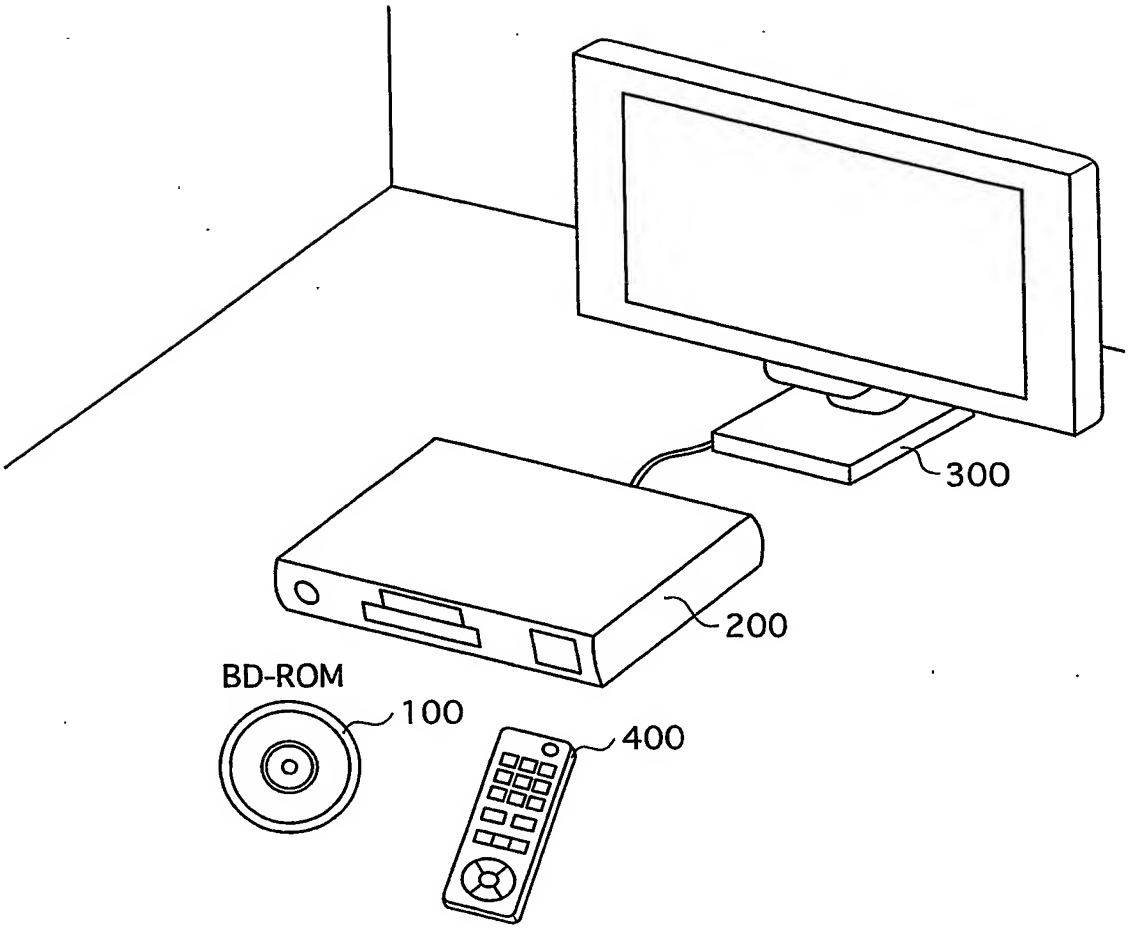


図2

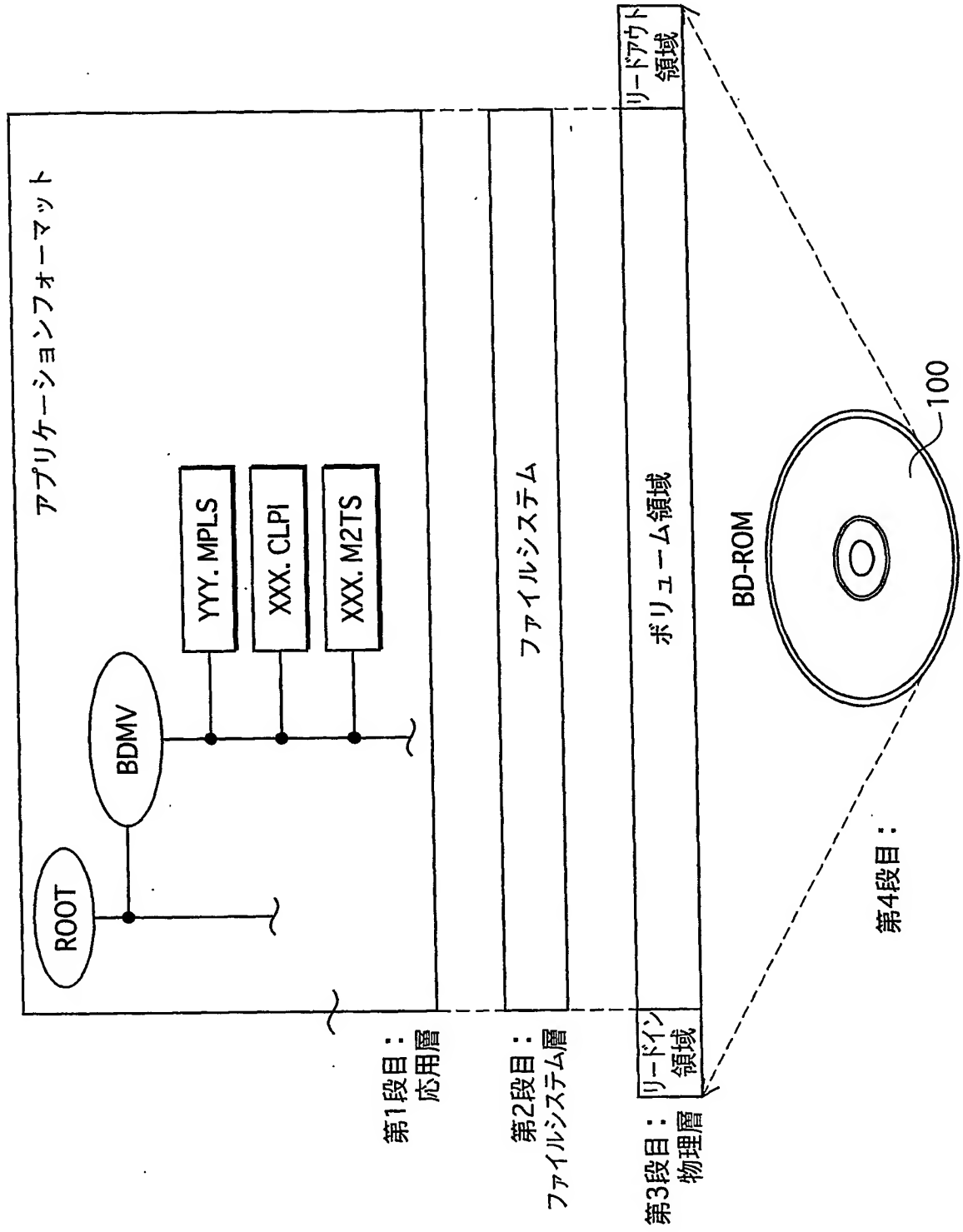

アプリケーションフォーマット
ROOT
BDMV
YYY. MPLS
XXX. CLPI
XXX. M2TS
ファイルシステム
ボリューム領域
リードアウト領域
リードイン領域
BD-ROM
100
第1段目：応用層
第2段目：ファイルシステム層
第3段目：物理層
第4段目：

図3

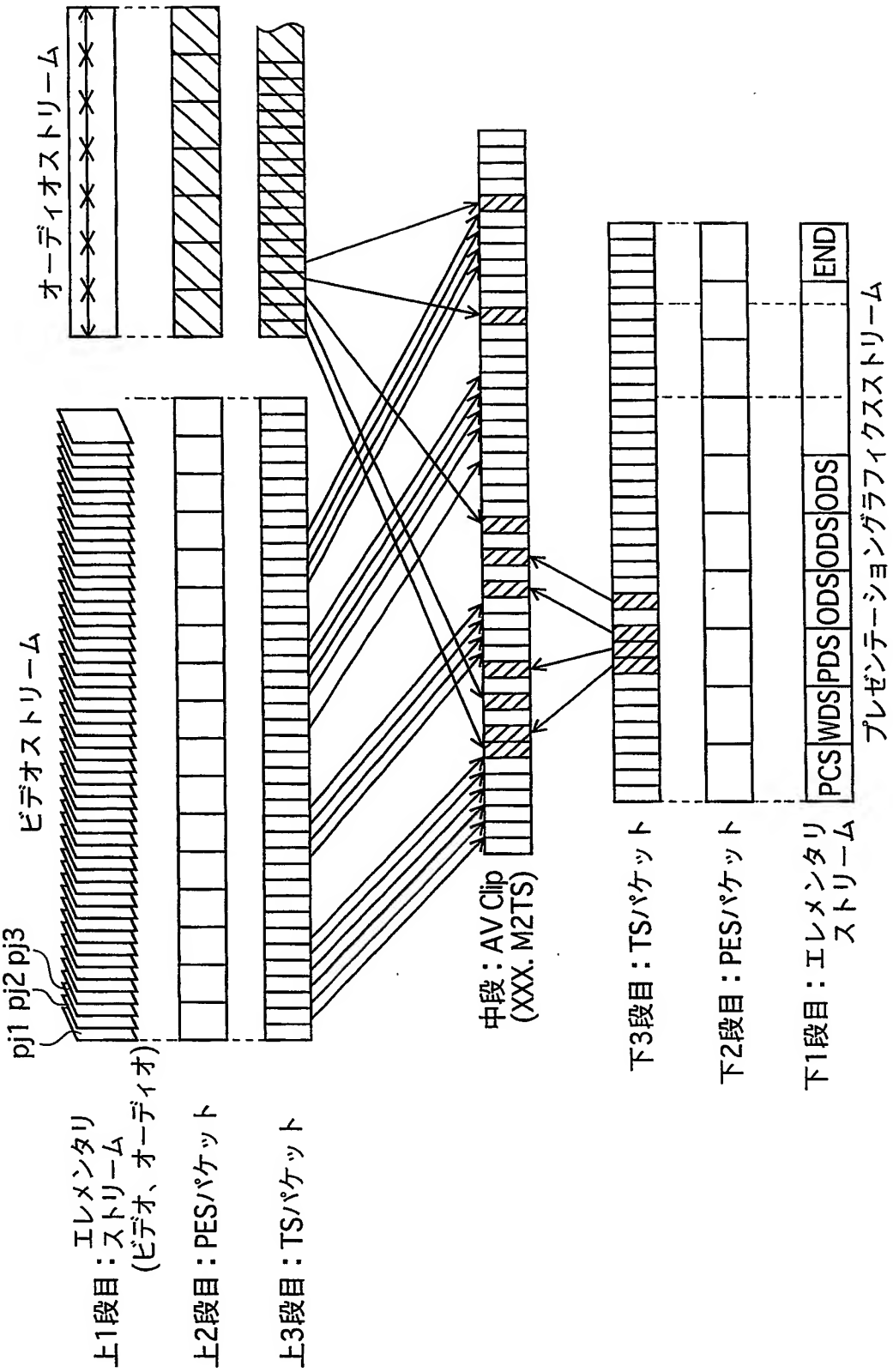

オーディオストリーム
ビデオストリーム
pj1 pj2 pj3
上1段目：エレメンタリストリーム（ビデオ、オーディオ）
上2段目：PESパケット
上3段目：TSパケット
中段：AV Clip (XXX. M2TS)
下3段目：TSパケット
下2段目：PESパケット
下1段目：エレメンタリストリーム
PCS
WDS
PDS
ODS
ODS
ODS
END
プレゼンテーショングラフィクスストリーム

図4

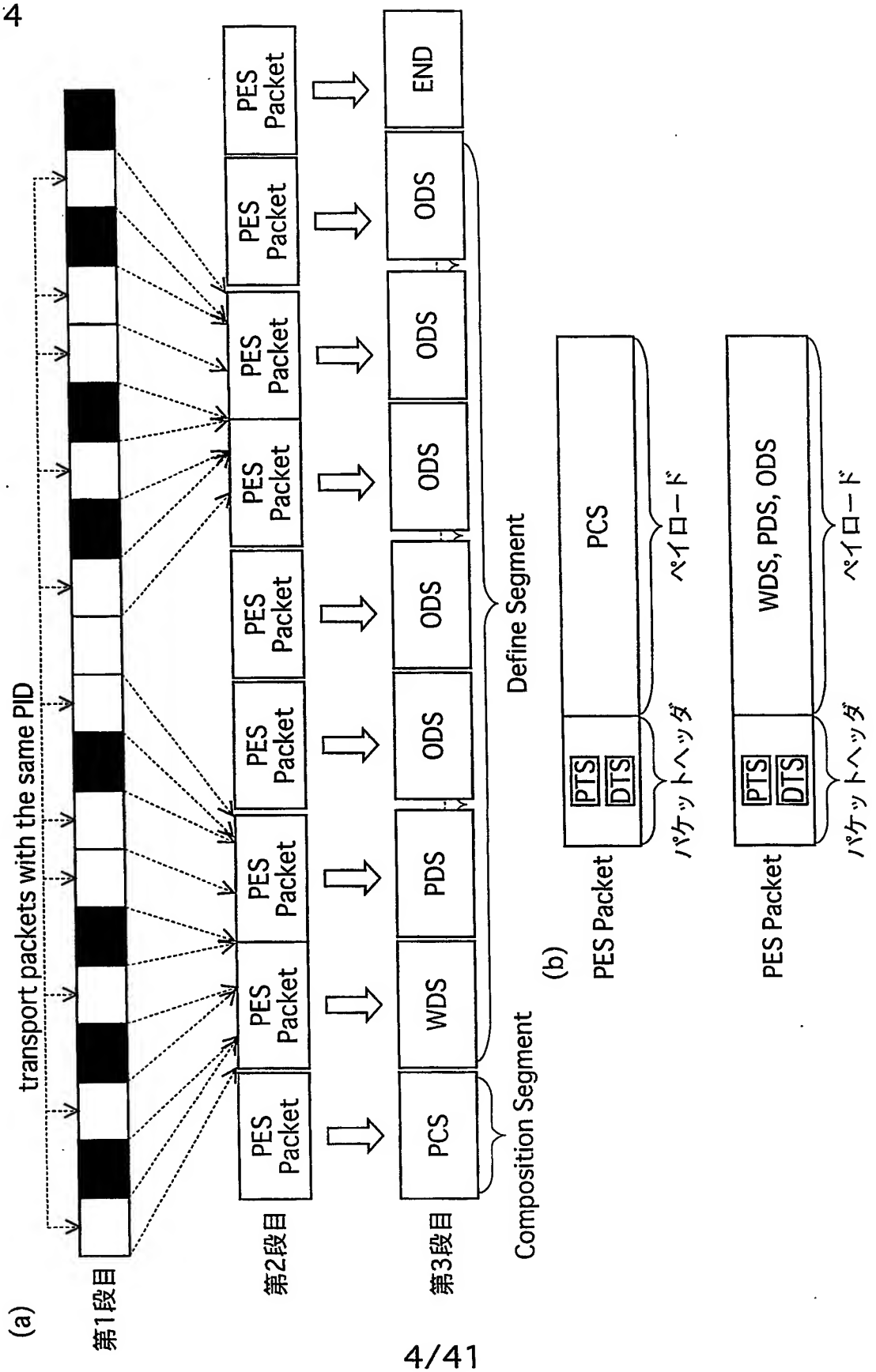

(a)
第1段目
transport packets with the same PID
第2段目
第3段目
PES Packet
PCS
WDS
PDS
ODS
END
Define Segment
Composition Segment
(b)
PES Packet
PTS
DTS
PCS
パケットヘッダ
ペイロード
WDS, PDS, ODS

図5

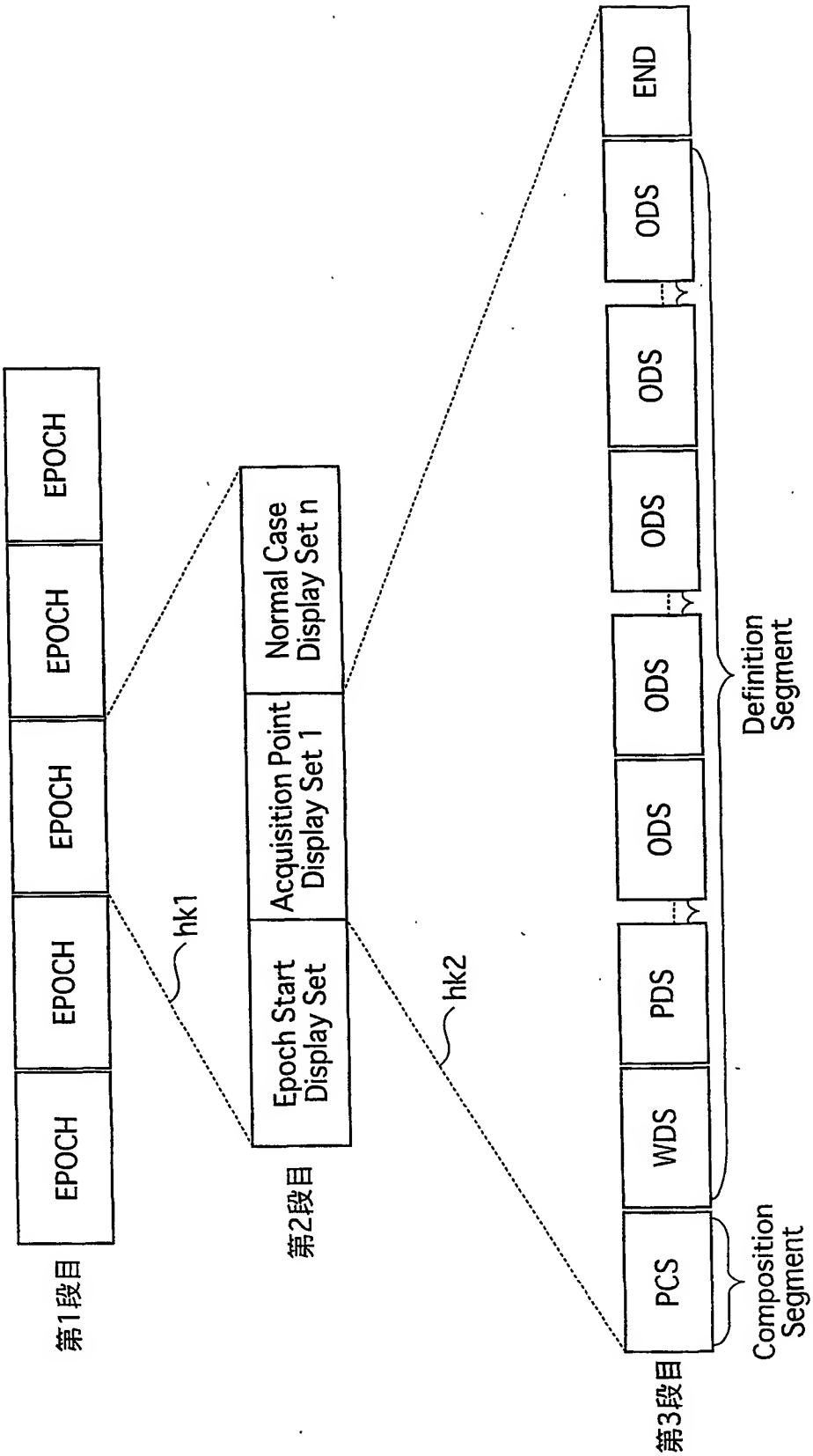

EPOCH
EPOCH
EPOCH
EPOCH
EPOCH
第1段目
hk1
Epoch Start Display Set
Acquisition Point Display Set 1
Normal Case Display Set n
第2段目
hk2
PCS
WDS
PDS
ODS
ODS
ODS
ODS
ODS
END
第3段目
Composition Segment
Definition Segment

図6

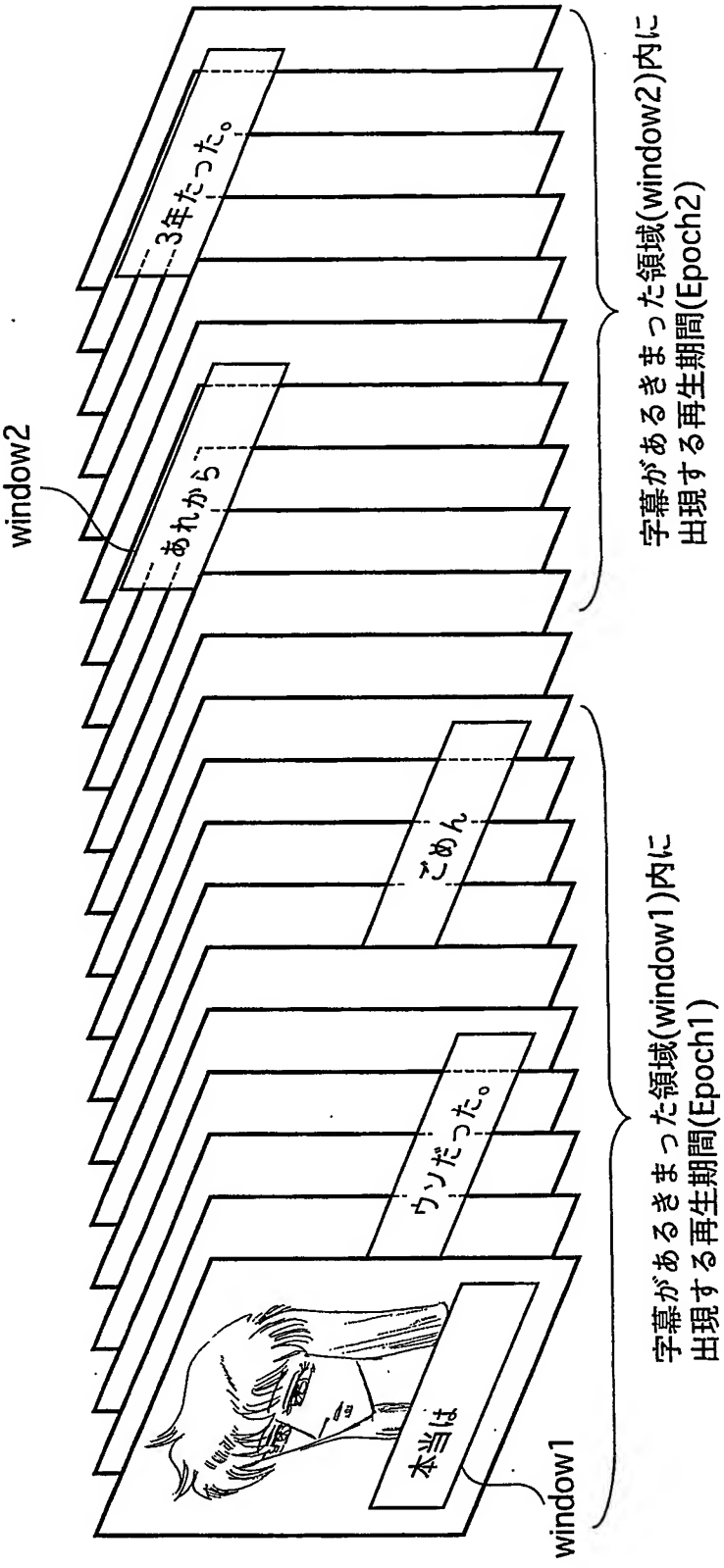

window2
あれから
3年たった。
ごめん
ウソだった。
本当は
window1
字幕があるきまった領域(window2)内に
出現する再生期間(Epoch2)
字幕があるきまった領域(window1)内に
出現する再生期間(Epoch1)

図7

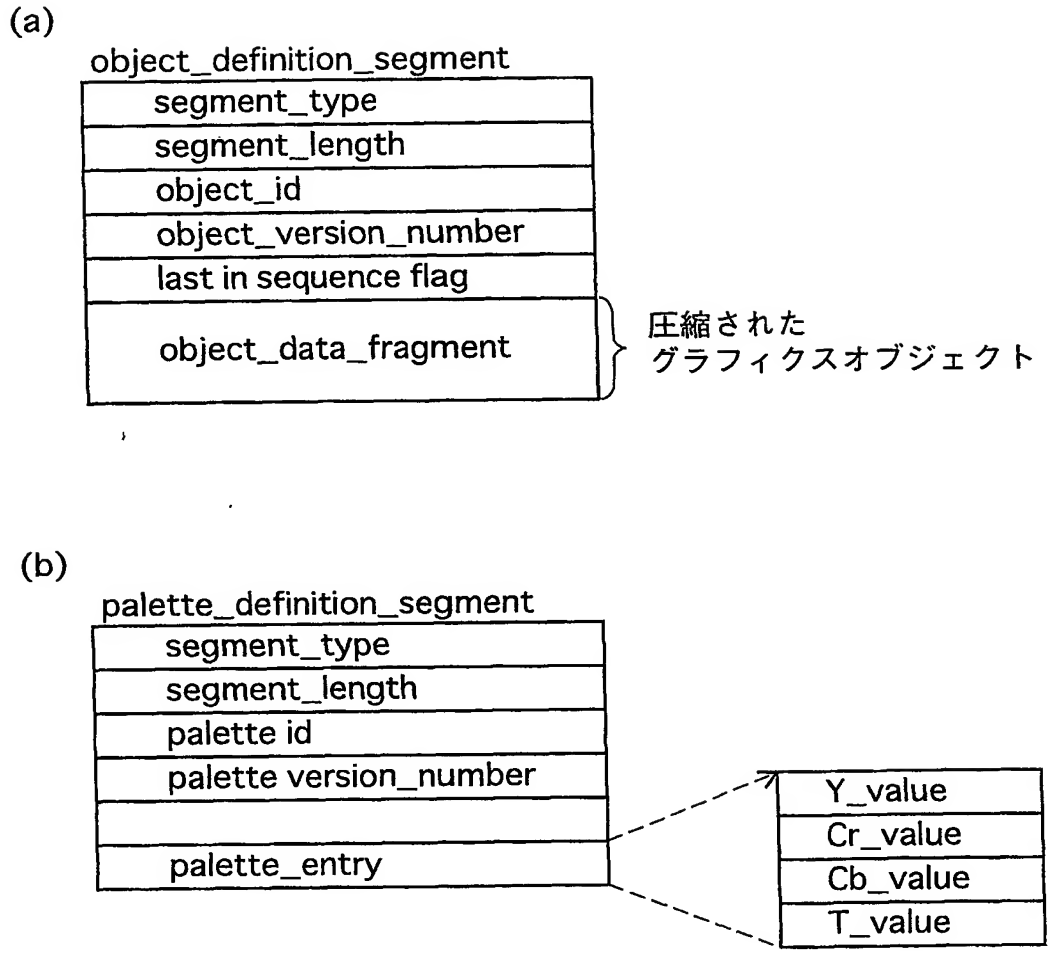

(a)
object_definition_segment
segment_type
segment_length
object_id
object_version_number
last in sequence flag
object_data_fragment
圧縮された
グラフィクスオブジェクト
(b)
palette_definition_segment
segment_type
segment_length
palette id
palette version_number
palette_entry
Y_value
Cr_value
Cb_value
T_value

図8

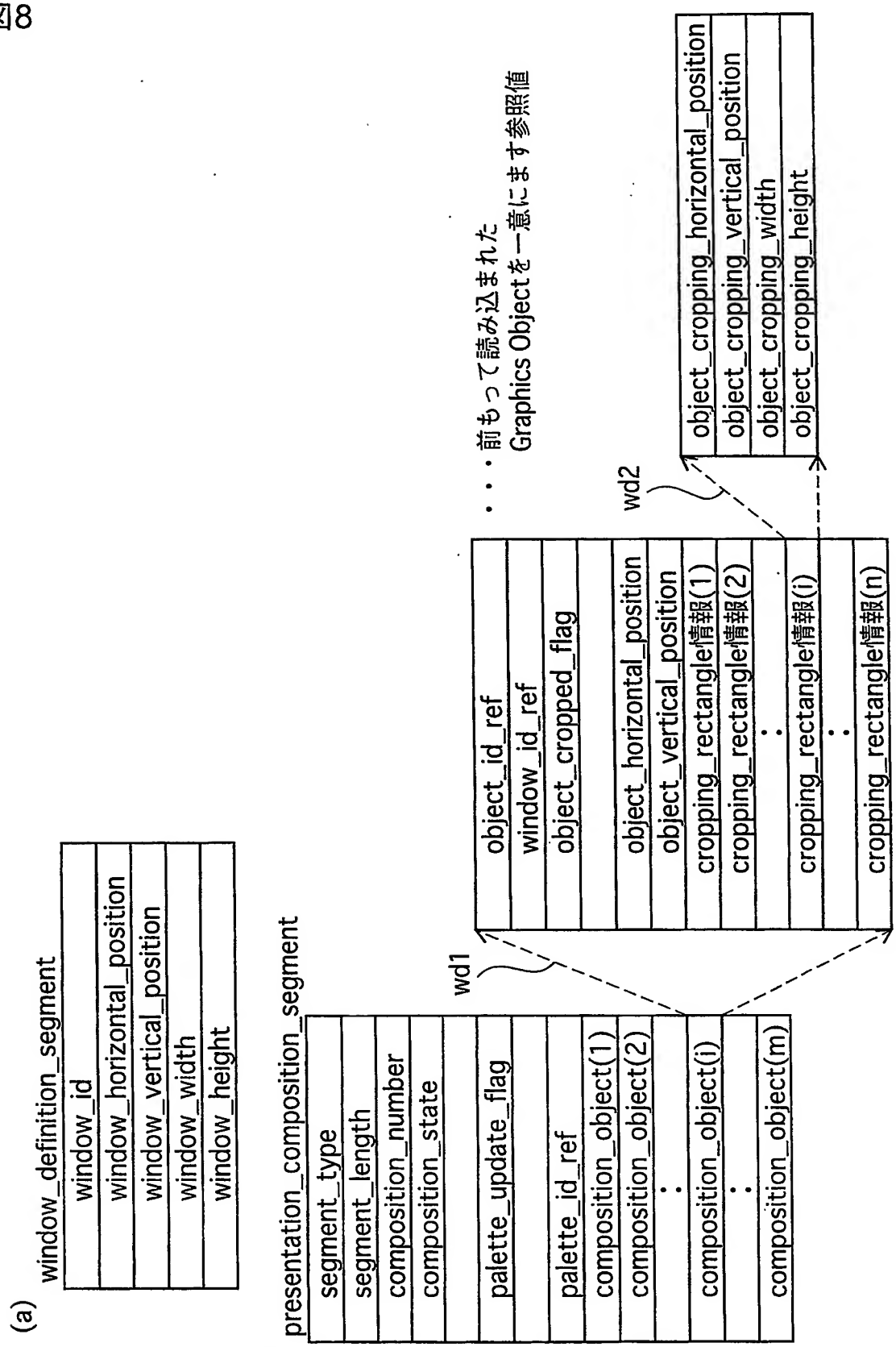

(a)
window_definition_segment
window_id
window_horizontal_position
window_vertical_position
window_width
window_height
(b)
presentation_composition_segment
segment_type
segment_length
composition_number
composition_state
palette_update_flag
palette_id_ref
composition_object(1)
composition_object(2)
composition_object(i)
composition_object(m)
wd1
object_id_ref
window_id_ref
object_cropped_flag
object_horizontal_position
object_vertical_position
cropping_rectangle情報(1)
cropping_rectangle情報(2)
cropping_rectangle情報(i)
cropping_rectangle情報(n)
・・・前もって読み込まれた
Graphics Objectを一意にます参照値
wd2
object_cropping_horizontal_position
object_cropping_vertical_position
object_cropping_width
object_cropping_height

図9

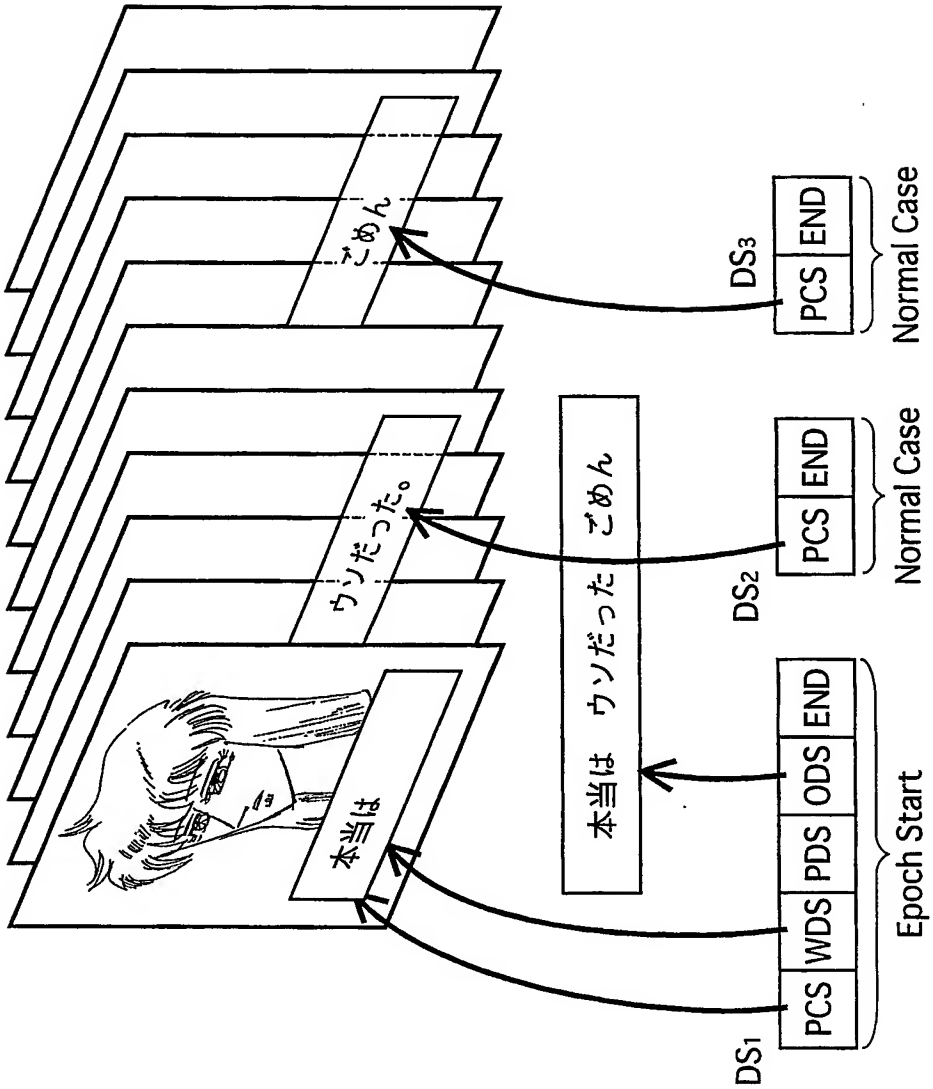

本当は
ウソだった。
ごめん
本当は ウソだった ごめん
DS1
PCS
WDS
PDS
ODS
END
Epoch Start
DS2
PCS
END
Normal Case
DS3
PCS
END
Normal Case

図10

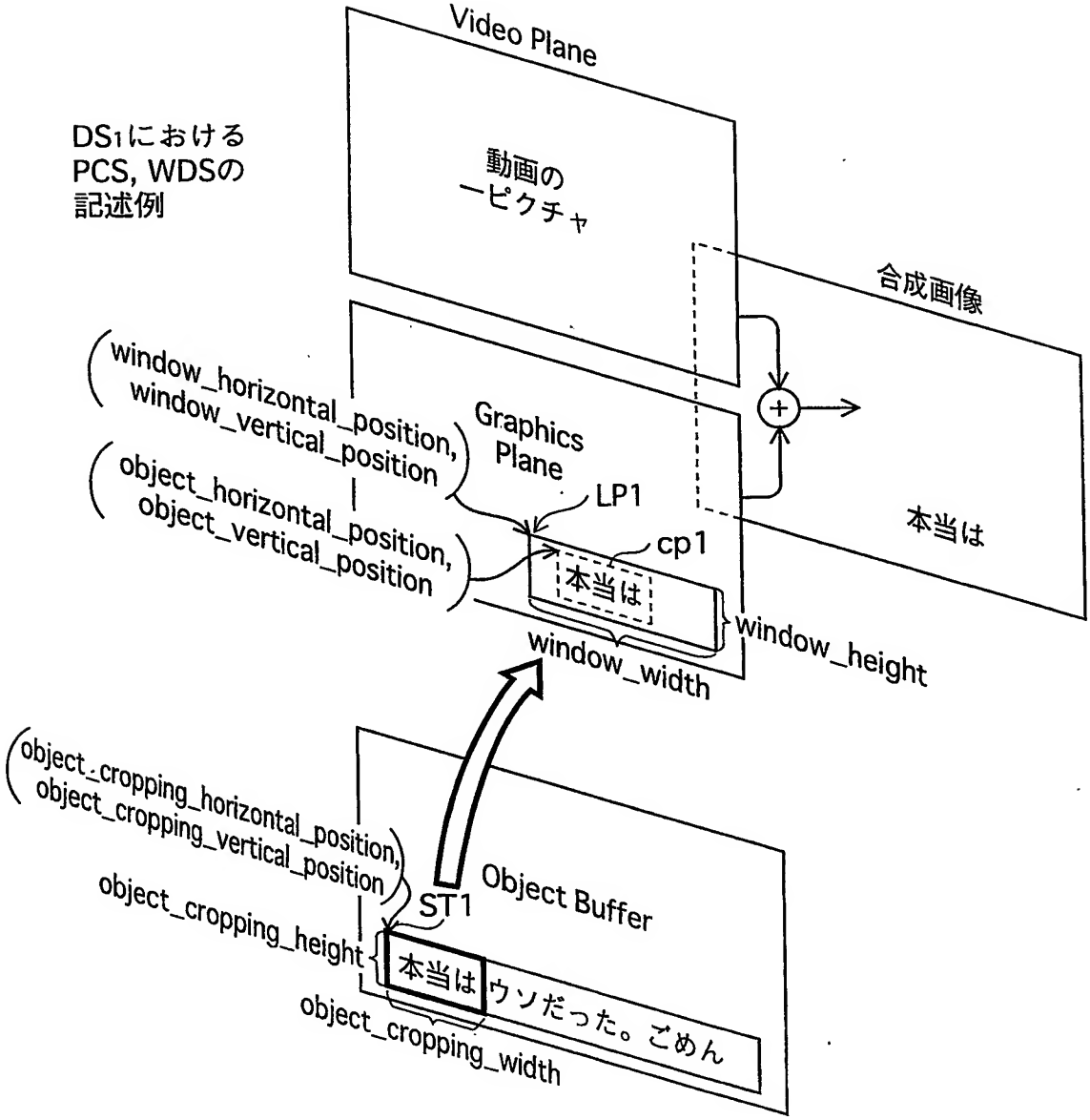

DS1における
PCS, WDSの
記述例
Video Plane
動画の
一ピクチャ
合成画像
window_horizontal_position,
window_vertical_position
Graphics
Plane
LP1
object_horizontal_position,
object_vertical_position
cp1
本当は
window_width
window_height
本当は
object_cropping_horizontal_position,
object_cropping_vertical_position
ST1
Object Buffer
object_cropping_height
本当は
ウソだった。ごめん
object_cropping_width

図11

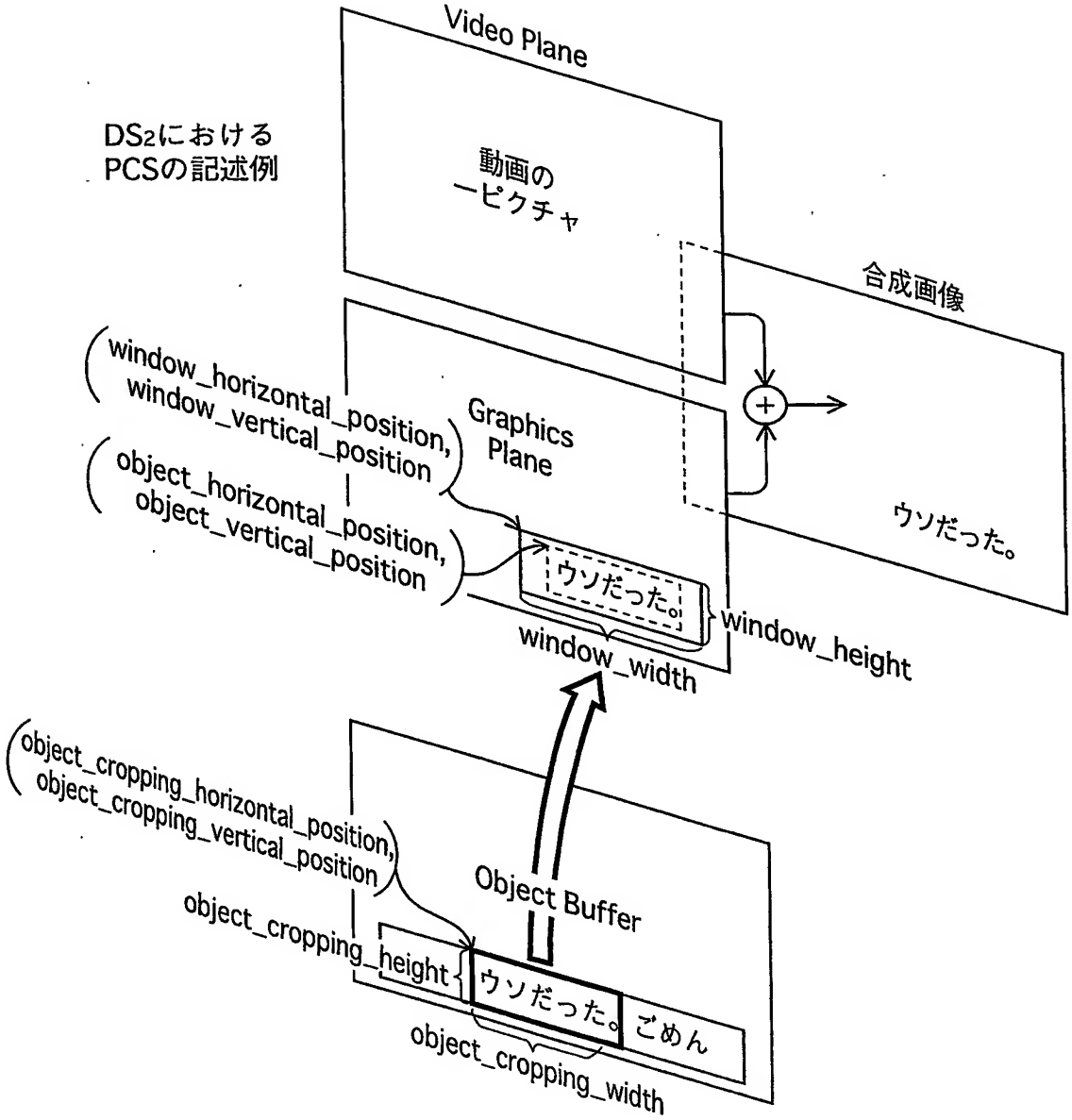

DS2における
PCSの記述例
Video Plane
動画の
一ピクチャ
合成画像
window_horizontal_position,
window_vertical_position
Graphics
Plane
object_horizontal_position,
object_vertical_position
ウソだった。
ウソだった。
window_height
window_width
object_cropping_horizontal_position,
object_cropping_vertical_position
Object Buffer
object_cropping_height
ウソだった。
ごめん
object_cropping_width

図12

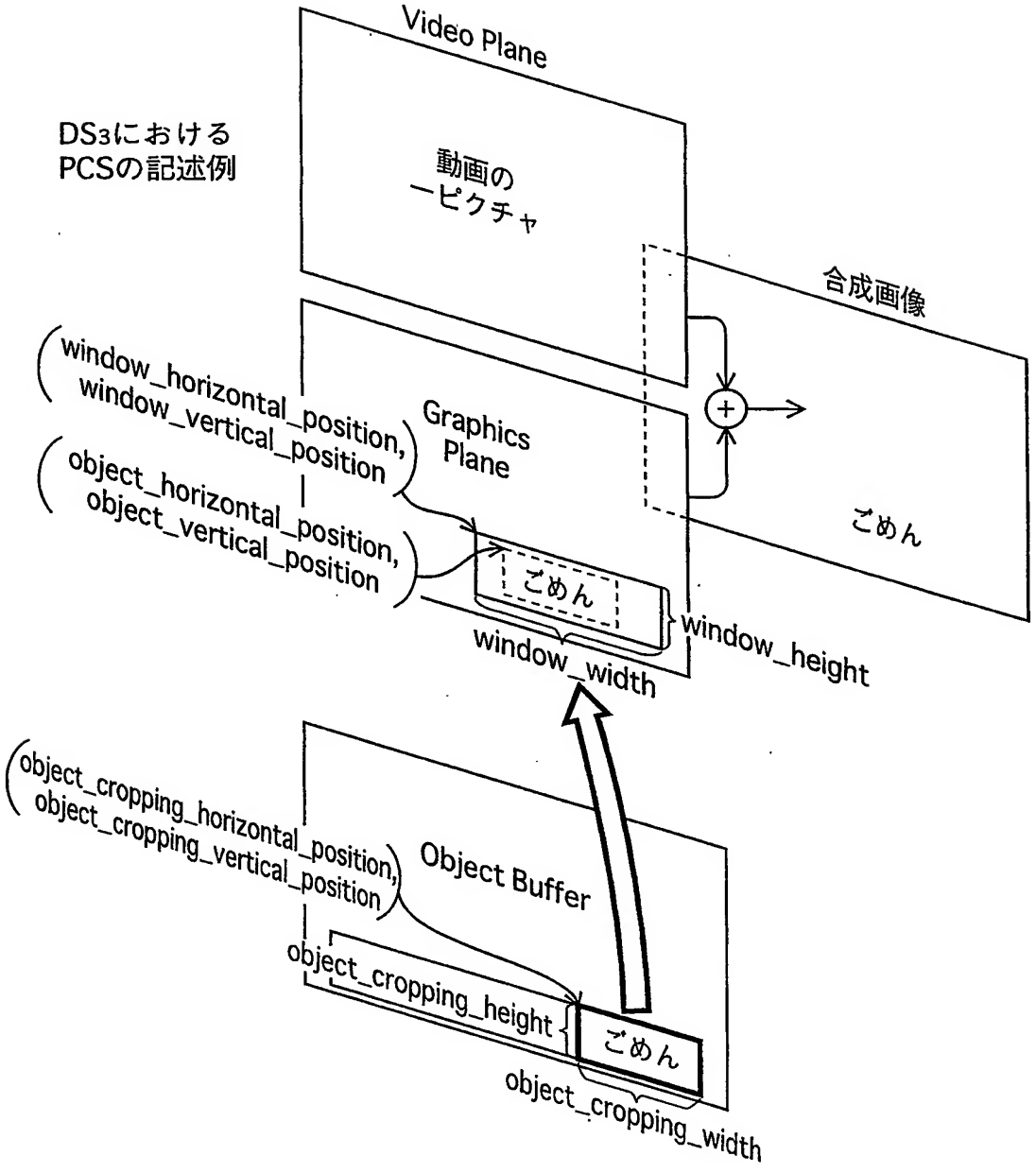

DS3における
PCSの記述例
Video Plane
動画の
一ピクチャ
合成画像
window_horizontal_position,
window_vertical_position
Graphics
Plane
object_horizontal_position,
object_vertical_position
ごめん
ごめん
window_height
window_width
object_cropping_horizontal_position,
object_cropping_vertical_position
Object Buffer
object_cropping_height
ごめん
object_cropping_width

図13

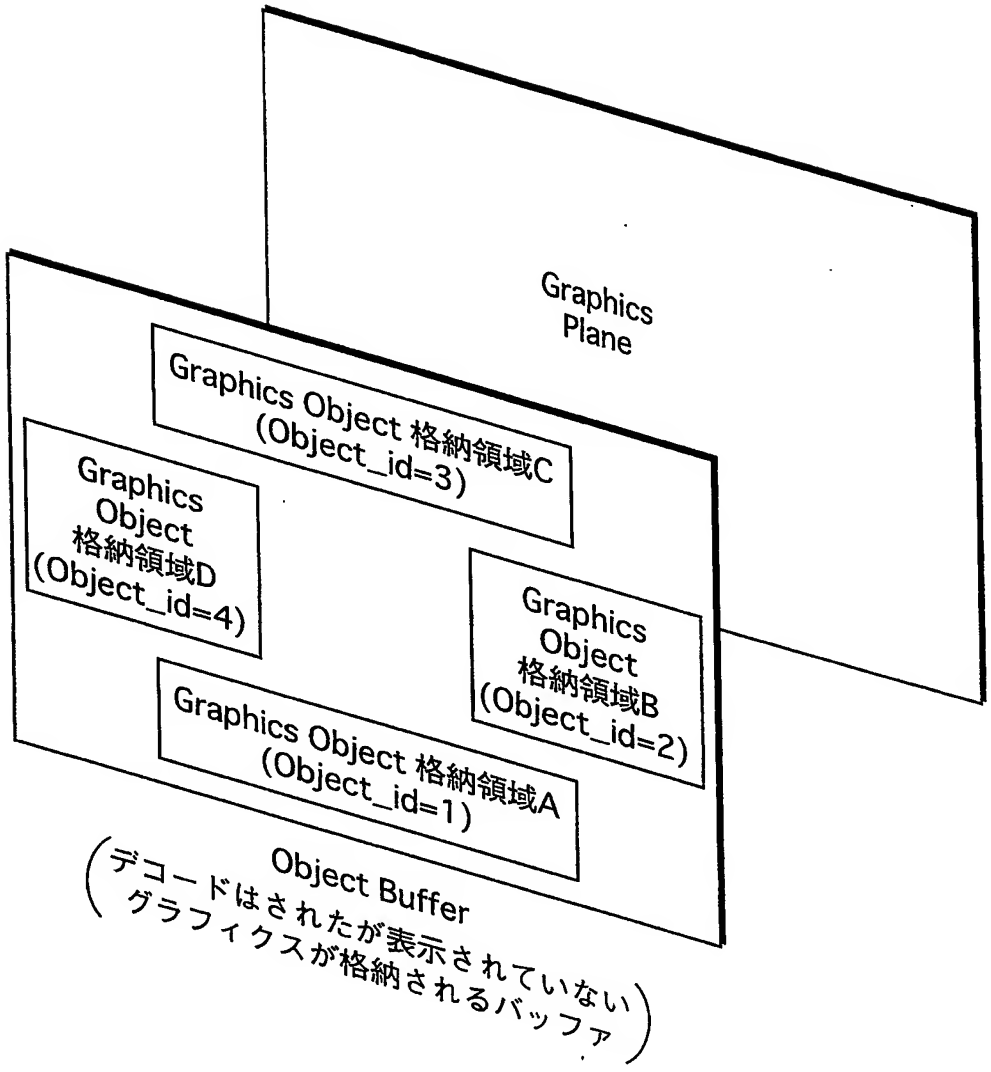

Graphics
Plane
Graphics Object 格納領域C
(Object_id=3)
Graphics
Object
格納領域D
(Object_id=4)
Graphics
Object
格納領域B
(Object_id=2)
Graphics Object 格納領域A
(Object_id=1)
Object Buffer
(デコードはされたが表示されていない
グラフィクスが格納されるバッファ)

図14

```
PTS( DSn[PCS)] )>=DTS( DSn[PCS] )+DECODEDURATION( DSn )
Where:
  • DECODEDURATION( DSn ) is calculated as follows:
decode_duration = 0 ;
decode_duration += PLANEINITIALIZATIONTIME( DSn ) ;
if( DSn. PCS. num_of_objects == 2 )
{
    decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
    if( DSn. PCS. OBJ[0]. window_id == DSn. PCS. OBJ[1]. window_id )
    {
            decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration ) ;
            decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*10^6 ) ;
    }
    else
    {
            decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*10^6 ) ;
            decode_duration += WAIT( DSn, DSn. PCS. OBJ[1], decode_duration );
            decode_duration += 90000*( SIZE( DSn. PCS. OBJ[1]. window_id )//256*10^6 ) ;
    }
}
else if( DSn. PCS. num_of_objects ==1 )
{
    decode_duration += WAIT( DSn, DSn. PCS. OBJ[0], decode_duration ) ;
    decode_duration += 90000*( SIZE( DSn. PCS. OBJ[0]. window_id )//256*10^6 ) ;
}
return decode_duration ;

  • PLANEINITIALIZATIONTIME( DSn ) is calculated as follows:
initialize_duration=0 ;
if( DSn. PCS. composition_state= = EPOCH_START )
{
  initialize_duration = 90000*( 8*video_width*video_height//256*10^6) ;
}
else
{
    for( i=0 ; i < WDS. num_windows ; i++ )
    {
            if( EMPTY(DSn.WDS.WIN[i],DSn ) )
                initialize_duration += 90000*( SIZE( DSn. WDS. WIN[i] )//256*10^6) ;
    }
}
return initialize_duration ;

  • WAIT( DSn, OBJ, current_duration ) is calculated as follows:
wait_duration = 0 ;
if( EXISTS( OBJ. object_id, DSn ) )
{
    object_definition_ready_time = PTS( GET( OBJ. object_id. DSn ) ) ;
    current_time = DTS( DSn. PCS )+current_duration ;
    if( current_time < object_definition_ready_time )
            wait_duration += object_definition_ready_time - current_time ) ;
}
return wait_duration ;
```

図15

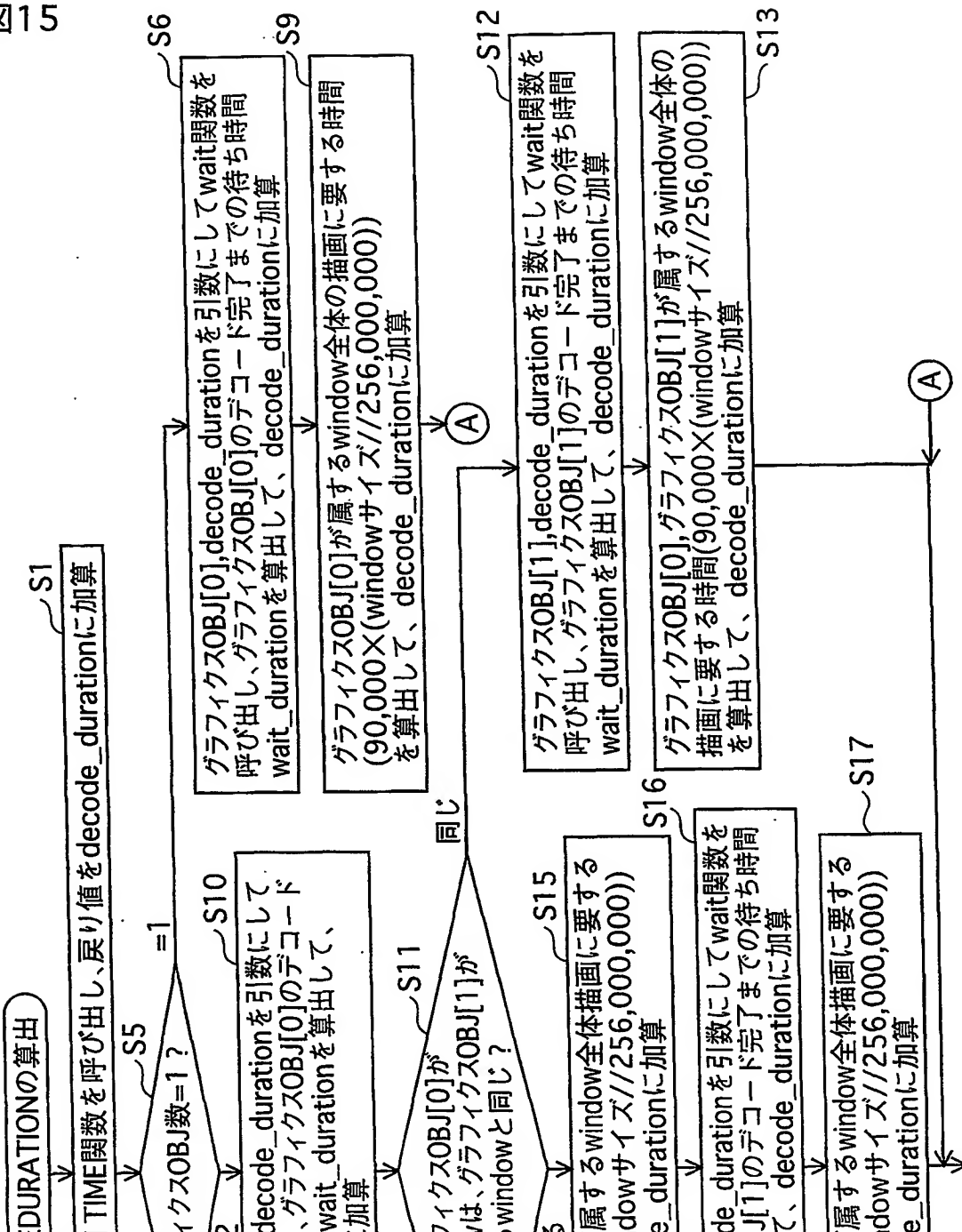
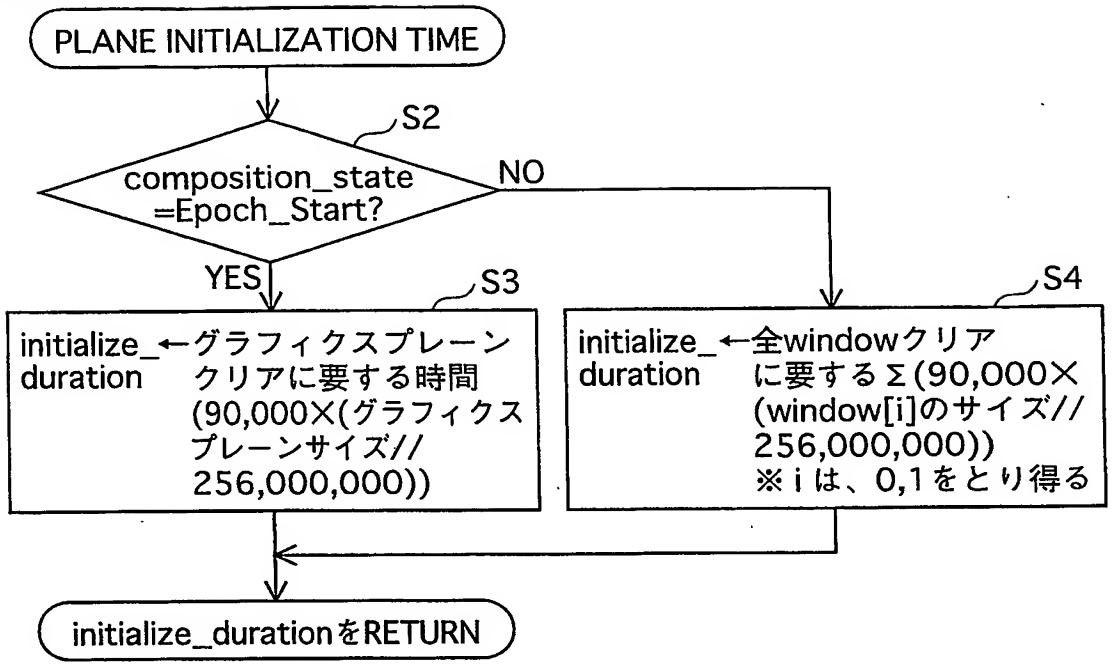

DECODEDURATIONの算出
S1
PLANE INITIALIZATION TIME関数を呼び出し、戻り値をdecode_durationに加算
S5
グラフィクスOBJ数=1？
=1
S6
グラフィクスOBJ[0],decode_durationを引数にしてwait関数を呼び出し、グラフィクスOBJ[0]のデコード完了までの待ち時間wait_durationを算出して、decode_durationに加算
S9
グラフィクスOBJ[0]が属するwindow全体の描画に要する時間(90,000×(windowサイズ//256,000,000))を算出して、decode_durationに加算
=2
S10
グラフィクスOBJ[0],decode_durationを引数にしてwait関数を呼び出し、グラフィクスOBJ[0]のデコード完了までの待ち時間wait_durationを算出して、decode_durationに加算
S11
グラフィクスOBJ[0]が属するwindowは、グラフィクスOBJ[1]が属するwindowと同じ？
同じ
A
S12
グラフィクスOBJ[1],decode_durationを引数にしてwait関数を呼び出し、グラフィクスOBJ[1]のデコード完了までの待ち時間wait_durationを算出して、decode_durationに加算
S13
グラフィクスOBJ[0],グラフィクスOBJ[1]が属するwindow全体の描画に要する時間(90,000×(windowサイズ//256,000,000))を算出して、decode_durationに加算
異なる
S15
グラフィクスOBJ[0]が属するwindow全体描画に要する時間(90,000×(windowサイズ//256,000,000))を算出して、decode_durationに加算
S16
グラフィクスOBJ[1],decode_durationを引数にしてwait関数を呼び出し、グラフィクスOBJ[1]のデコード完了までの待ち時間wait_durationを算出して、decode_durationに加算
S17
グラフィクスOBJ[1]が属するwindow全体描画に要する時間(90,000×(windowサイズ//256,000,000))を算出して、decode_durationに加算
A
END

図16

(a)


PLANE INITIALIZATION TIME
S2
composition_state =Epoch_Start?
NO
YES
S3
initialize_duration
←グラフィクスプレーンクリアに要する時間 (90,000×(グラフィクスプレーンサイズ// 256,000,000))
S4
initialize_duration
←全windowクリアに要するΣ(90,000×(window[i]のサイズ// 256,000,000)) ※iは、0,1をとり得る
initialize_durationをRETURN

(b)

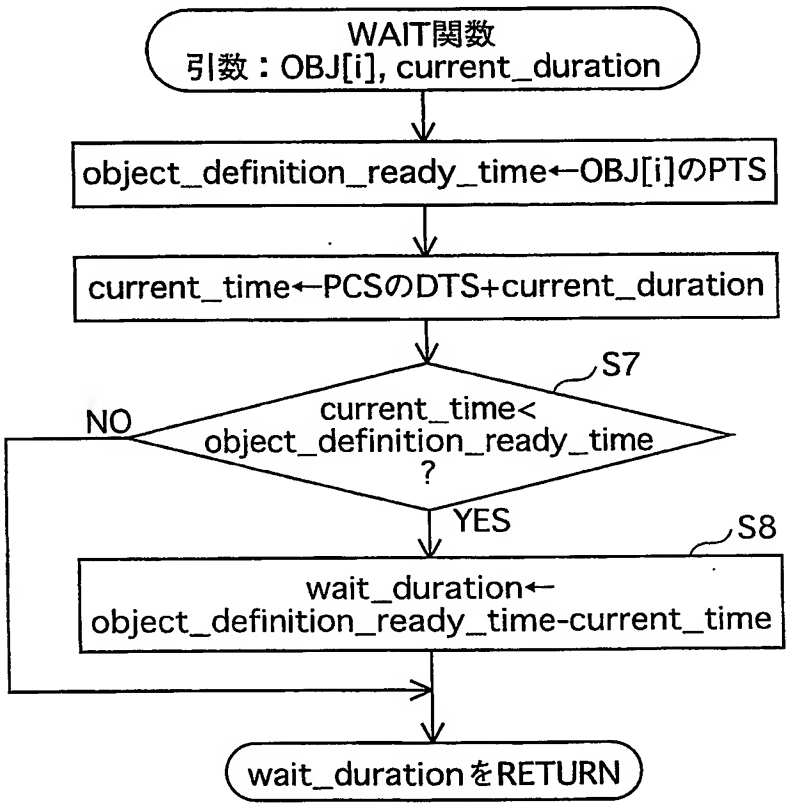

WAIT関数
引数：OBJ[i], current_duration
object_definition_ready_time←OBJ[i]のPTS
current_time←PCSのDTS+current_duration
S7
current_time< object_definition_ready_time ?
NO
YES
S8
wait_duration← object_definition_ready_time-current_time
wait_durationをRETURN

図17

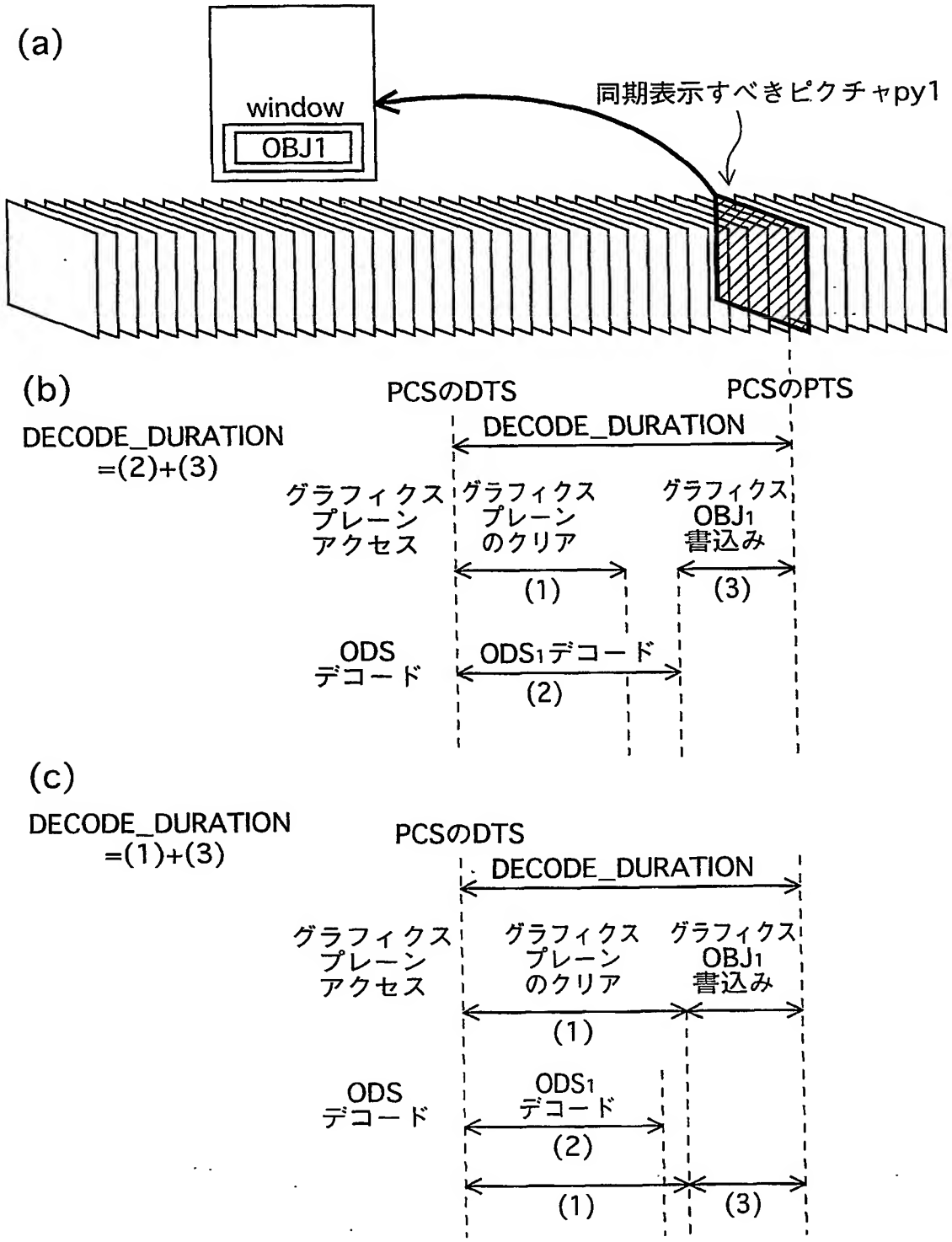
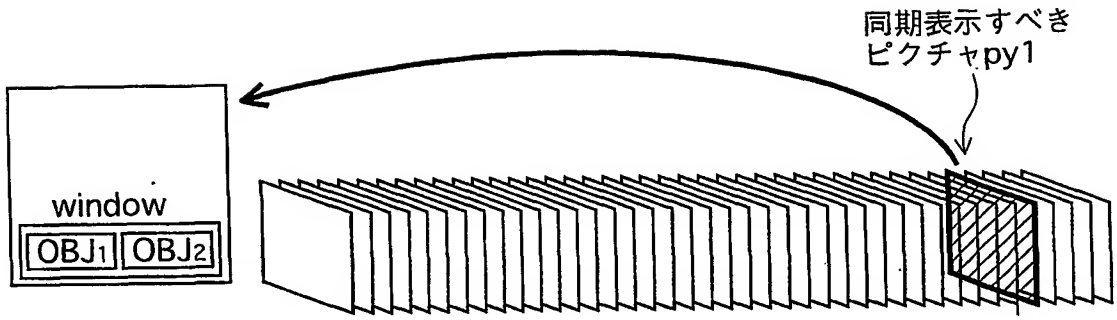
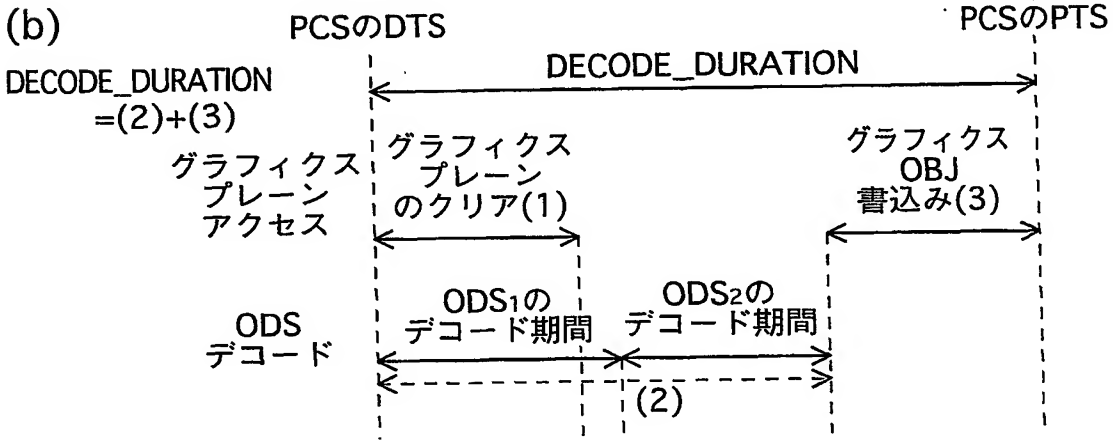

(a)
window
OBJ1
同期表示すべきピクチャpy1
(b)
PCSのDTS
PCSのPTS
DECODE_DURATION
=(2)+(3)
DECODE_DURATION
グラフィクス
プレーン
アクセス
グラフィクス
プレーン
のクリア
グラフィクス
OBJ1
書込み
(1)
(3)
ODS
デコード
ODS1デコード
(2)
(c)
DECODE_DURATION
=(1)+(3)
PCSのDTS
DECODE_DURATION
グラフィクス
プレーン
アクセス
グラフィクス
プレーン
のクリア
グラフィクス
OBJ1
書込み
(1)
ODS
デコード
ODS1
デコード
(2)
(1)
(3)

図18

(a)


同期表示すべき
ピクチャpy1
window
OBJ1
OBJ2

(b)


PCSのDTS
PCSのPTS
DECODE_DURATION
DECODE_DURATION
=(2)+(3)
グラフィクス
プレーン
アクセス
グラフィクス
プレーン
のクリア(1)
グラフィクス
OBJ
書込み(3)
ODS
デコード
ODS1の
デコード期間
ODS2の
デコード期間
(2)

(c)

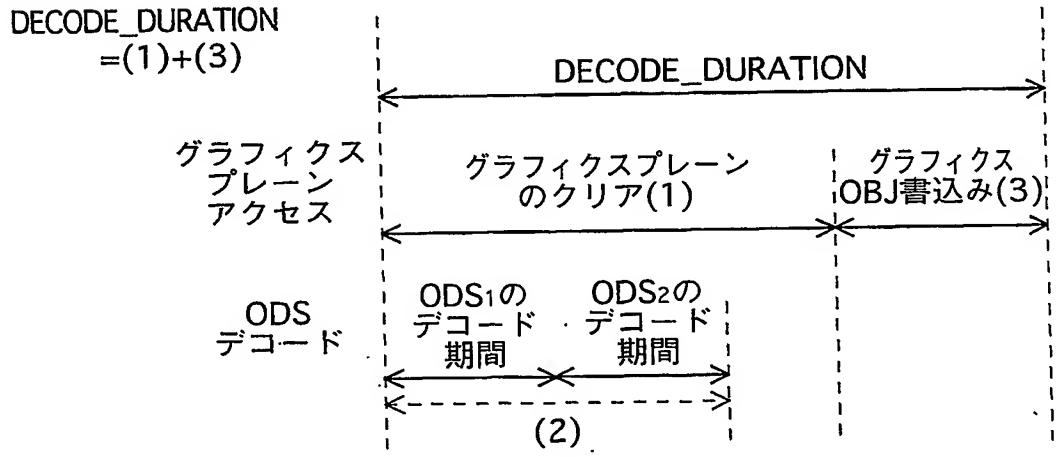

DECODE_DURATION
=(1)+(3)
DECODE_DURATION
グラフィクス
プレーン
アクセス
グラフィクスプレーン
のクリア(1)
グラフィクス
OBJ書込み(3)
ODS
デコード
ODS1の
デコード
期間
ODS2の
デコード
期間
(2)

図19

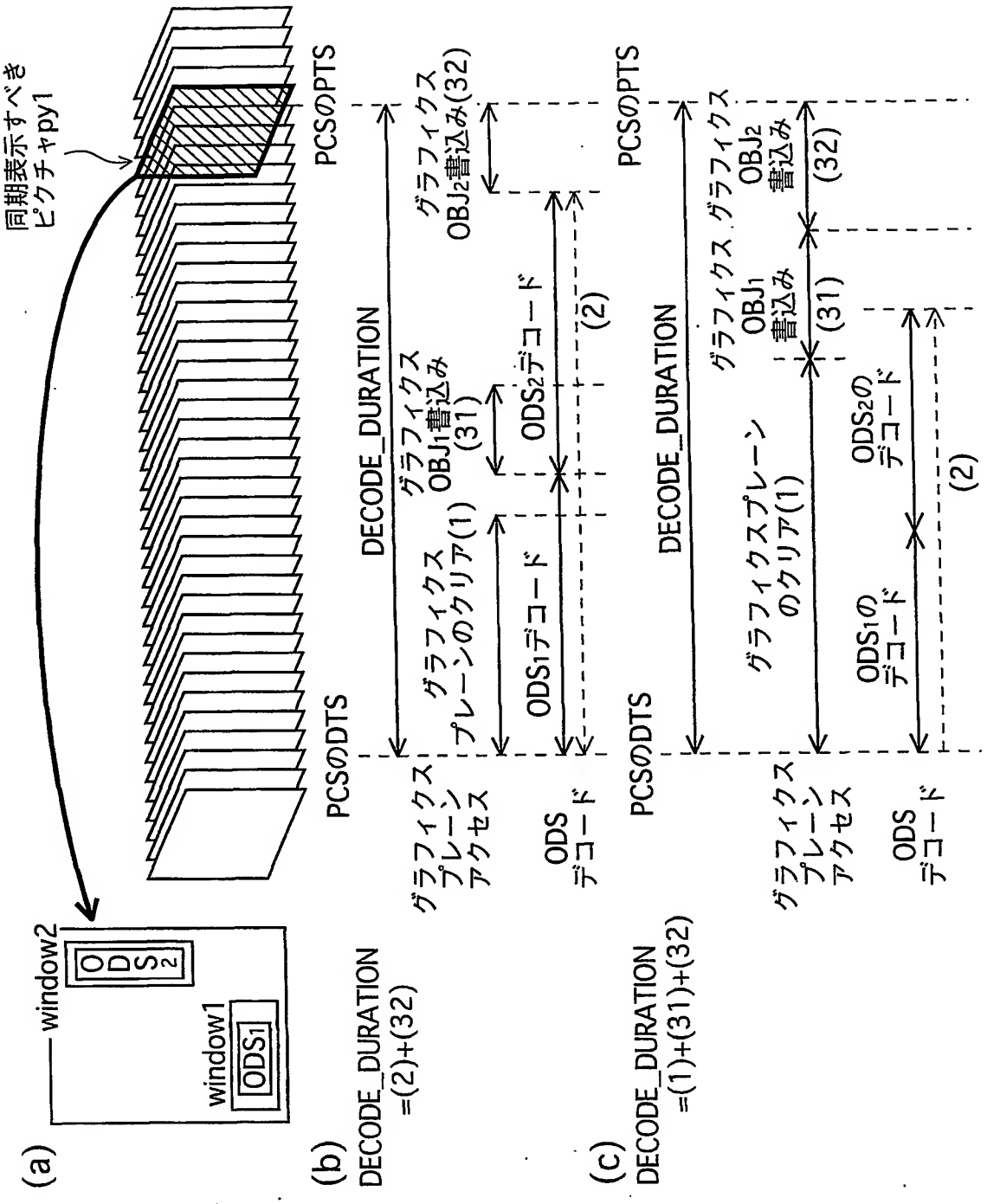

同期表示すべき
ピクチャpy1
window2
ODS2
window1
ODS1
(a)
(b)
DECODE_DURATION
=(2)+(32)
PCSのPTS
PCSのDTS
グラフィクス
プレーン
アクセス
グラフィクス
プレーンのクリア(1)
グラフィクス
OBJ1書込み
(31)
ODS1デコード
ODS
デコード
グラフィクス
OBJ2書込み(32)
ODS2デコード
(2)
(c)
DECODE_DURATION
=(1)+(31)+(32)
PCSのPTS
PCSのDTS
グラフィクス
プレーン
アクセス
グラフィクス
プレーンスプレーン
のクリア(1)
グラフィクス
OBJ1
書込み
(31)
ODS1の
デコード
ODS
デコード
グラフィクス
OBJ2
書込み
(32)
ODS2の
デコード
(2)

図20

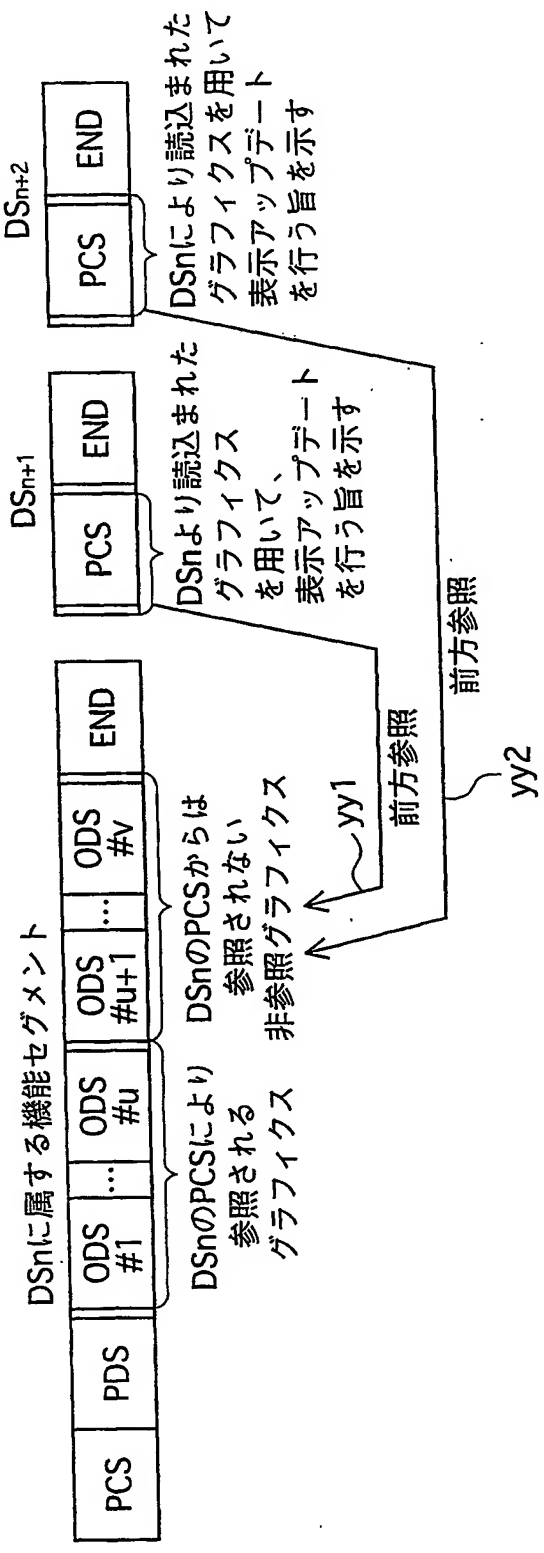

DSnに属する機能セグメント
PCS
PDS
ODS #1
…
ODS #u
ODS #u+1
…
ODS #v
END
DSnのPCSにより参照されるグラフィクス
DSnのPCSからは参照されない非参照グラフィクス
DSn+1
PCS
END
DSnより読込まれたグラフィクスを用いて、表示アップデートを行う旨を示す
DSn+2
PCS
END
DSnにより読込まれたグラフィクスを用いて表示アップデートを行う旨を示す
yy1
前方参照
前方参照
yy2

図21

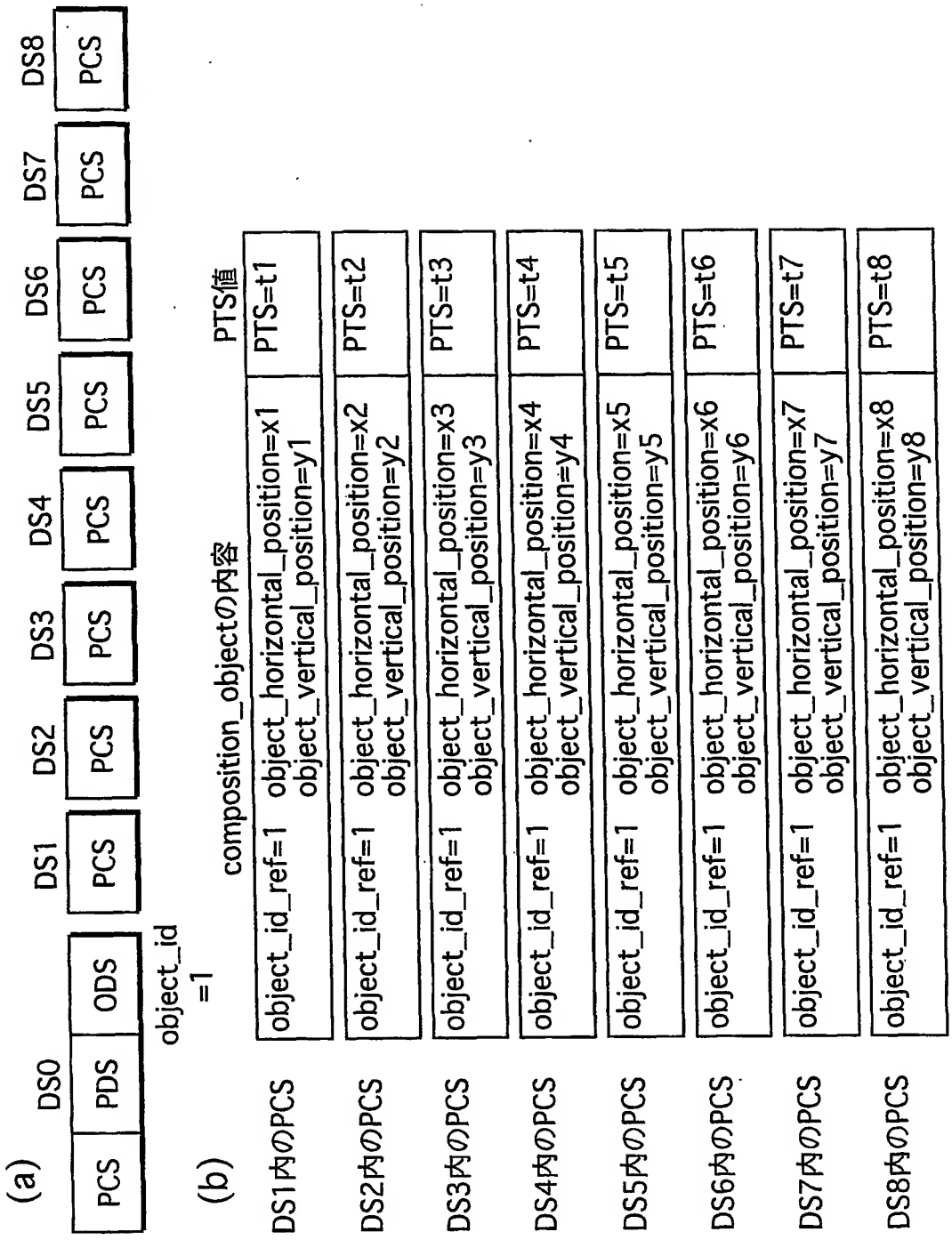

(a)
DS0
PCS
PDS
ODS
DS1
PCS
DS2
PCS
DS3
PCS
DS4
PCS
DS5
PCS
DS6
PCS
DS7
PCS
DS8
PCS
(b)
object_id =1
composition_objectの内容
PTS値
DS1内のPCS
object_id_ref=1
object_horizontal_position=x1
object_vertical_position=y1
PTS=t1
DS2内のPCS
object_id_ref=1
object_horizontal_position=x2
object_vertical_position=y2
PTS=t2
DS3内のPCS
object_id_ref=1
object_horizontal_position=x3
object_vertical_position=y3
PTS=t3
DS4内のPCS
object_id_ref=1
object_horizontal_position=x4
object_vertical_position=y4
PTS=t4
DS5内のPCS
object_id_ref=1
object_horizontal_position=x5
object_vertical_position=y5
PTS=t5
DS6内のPCS
object_id_ref=1
object_horizontal_position=x6
object_vertical_position=y6
PTS=t6
DS7内のPCS
object_id_ref=1
object_horizontal_position=x7
object_vertical_position=y7
PTS=t7
DS8内のPCS
object_id_ref=1
object_horizontal_position=x8
object_vertical_position=y8
PTS=t8

図22

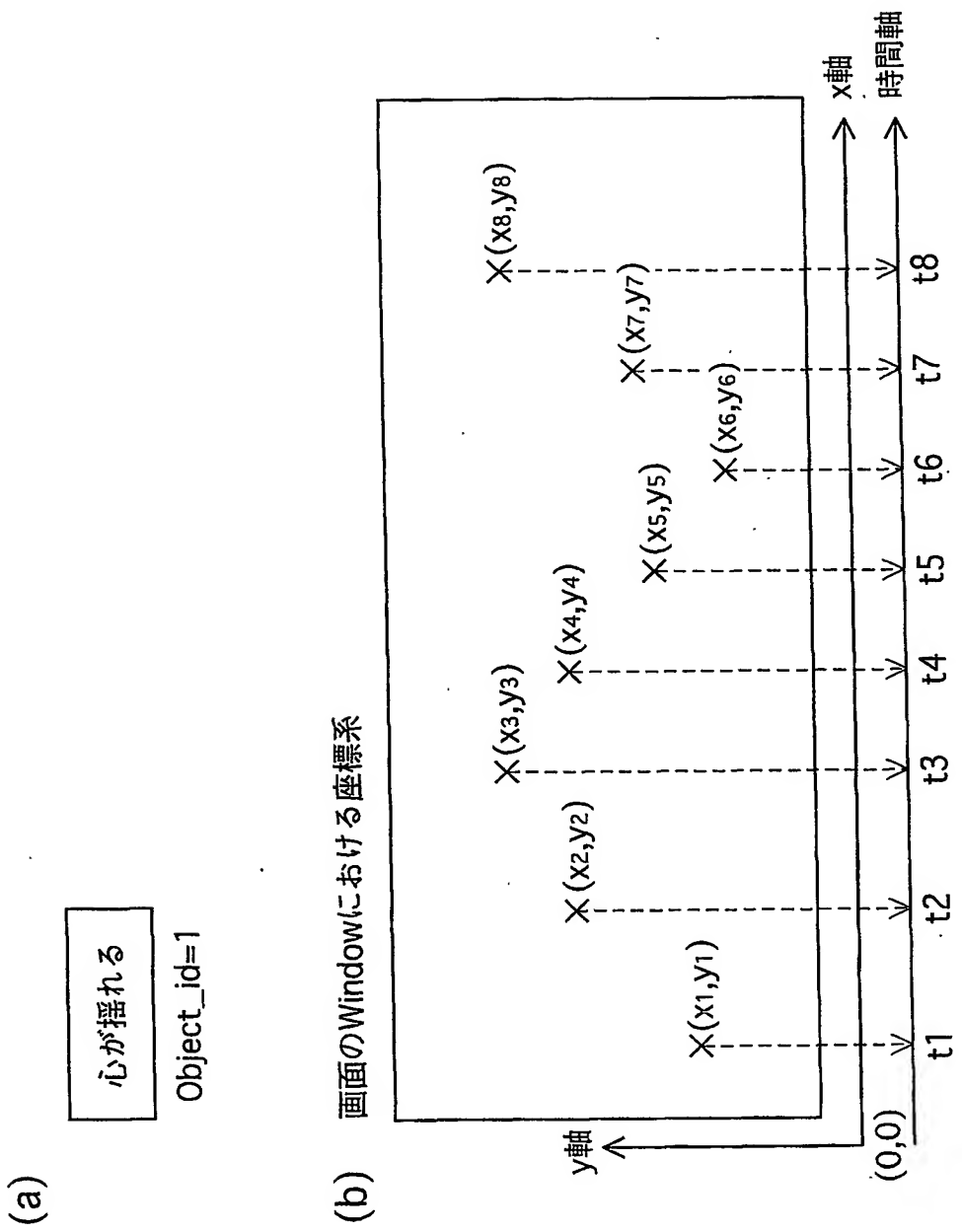

(a)
心が揺れる
Object_id=1
(b)
画面のWindowにおける座標系
y軸
x軸
時間軸
(0,0)
(x1,y1)
(x2,y2)
(x3,y3)
(x4,y4)
(x5,y5)
(x6,y6)
(x7,y7)
(x8,y8)
t1
t2
t3
t4
t5
t6
t7
t8

図23

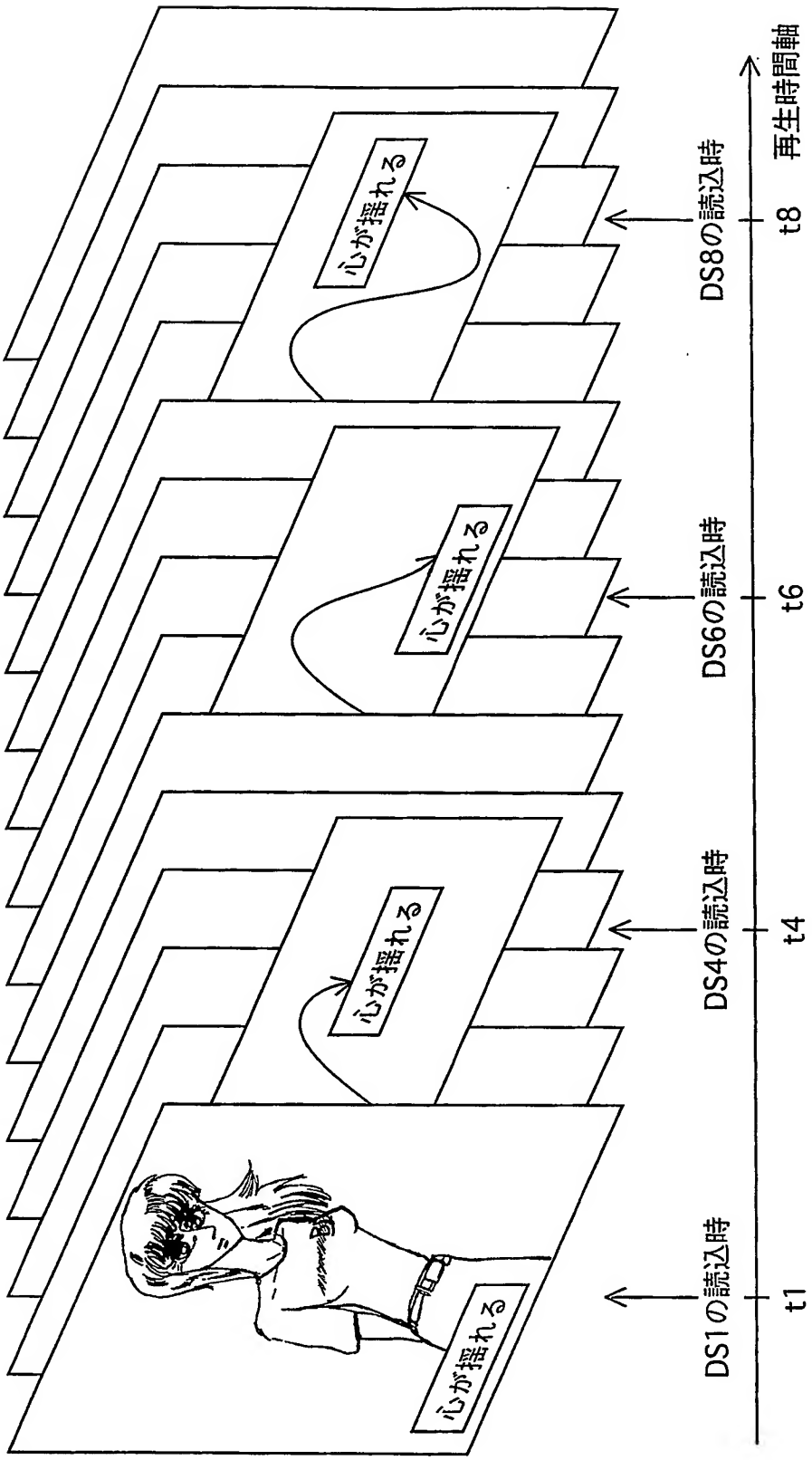

心が揺れる
心が揺れる
心が揺れる
心が揺れる
DS1の読込時
DS4の読込時
DS6の読込時
DS8の読込時
t1
t4
t6
t8
再生時間軸

図24

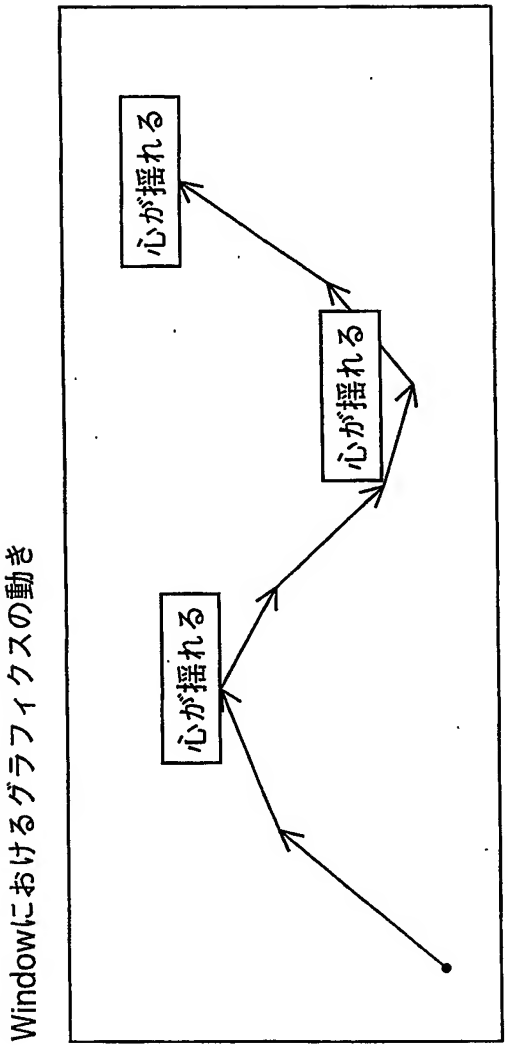


図25

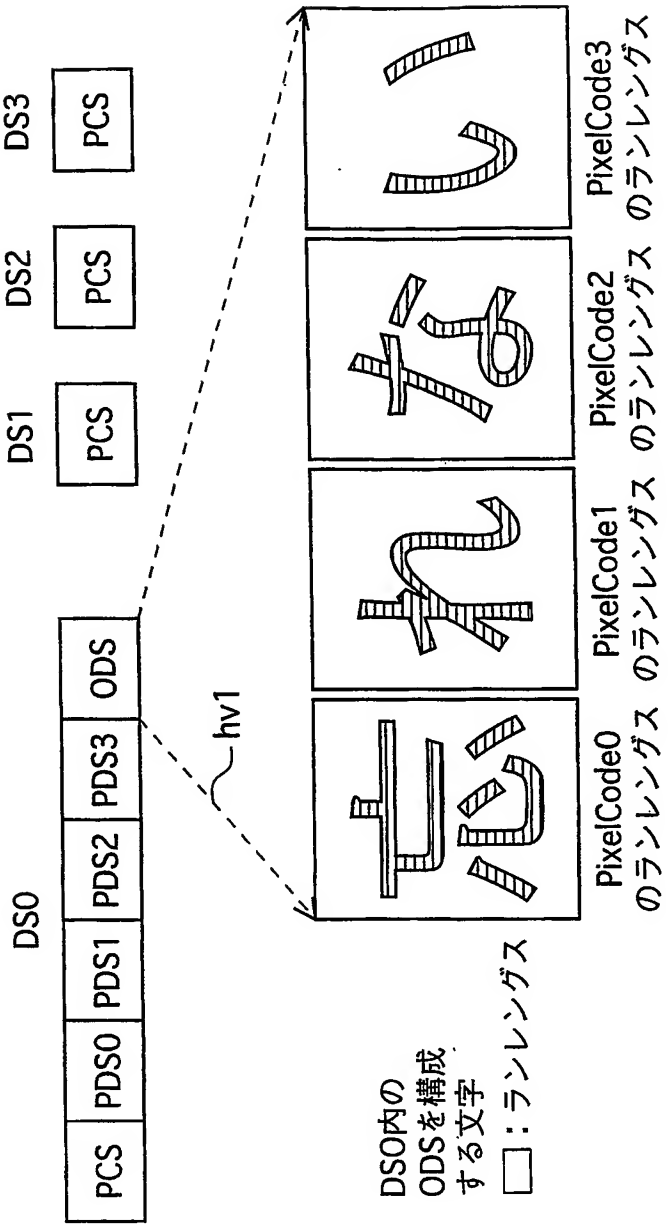

DS0
PCS
PDS0
PDS1
PDS2
PDS3
ODS
DS1
PCS
DS2
PCS
DS3
PCS
hv1
PixelCode0
のランレングス
PixelCode1
のランレングス
PixelCode2
のランレングス
PixelCode3
のランレングス
DSO内の
ODSを構成
する文字
□：ランレングス

図26

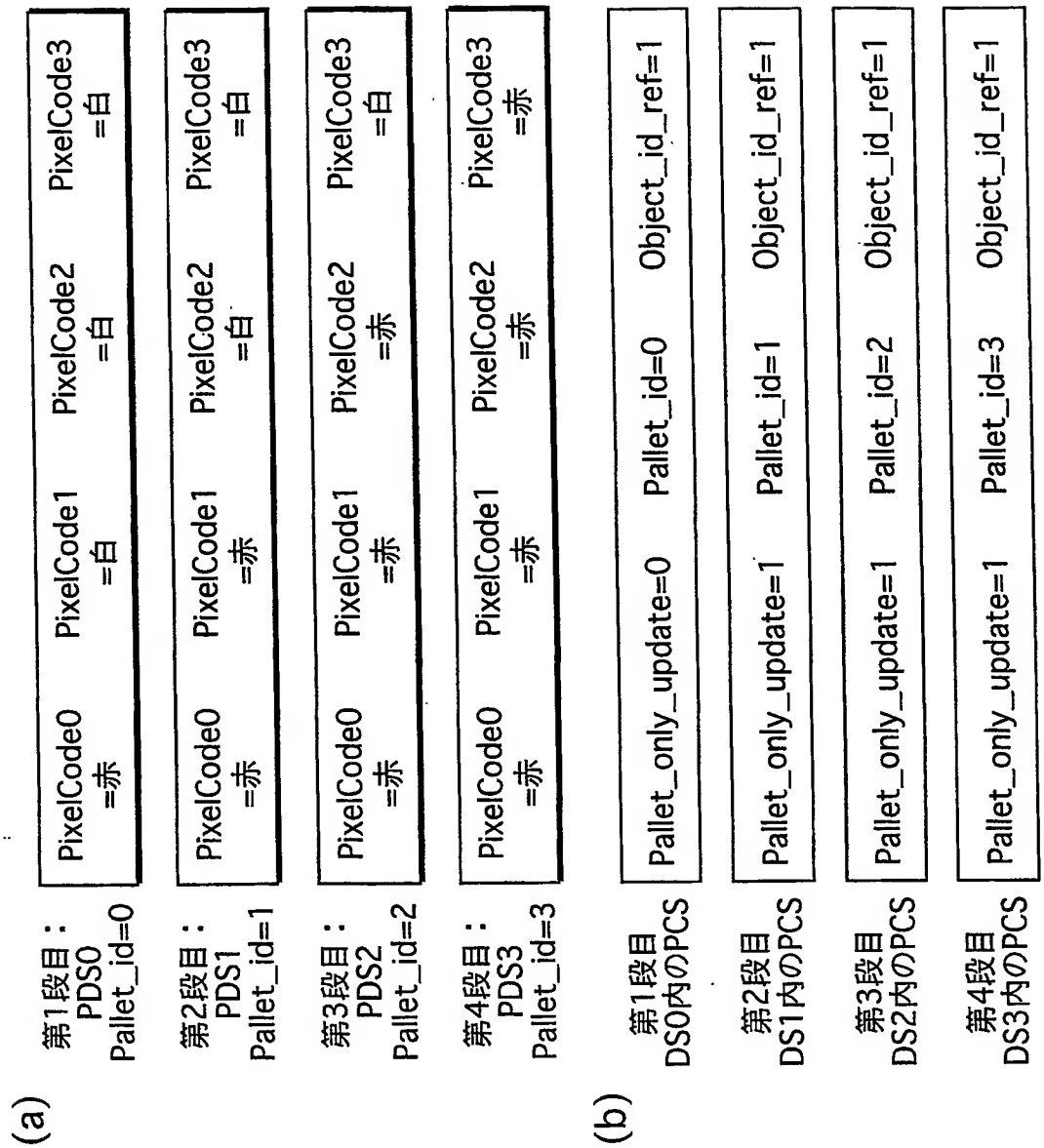

(a)
第1段目：PDS0 Pallet_id=0
PixelCode0 =赤
PixelCode1 =白
PixelCode2 =白
PixelCode3 =白
第2段目：PDS1 Pallet_id=1
PixelCode0 =赤
PixelCode1 =赤
PixelCode2 =白
PixelCode3 =白
第3段目：PDS2 Pallet_id=2
PixelCode0 =赤
PixelCode1 =赤
PixelCode2 =赤
PixelCode3 =白
第4段目：PDS3 Pallet_id=3
PixelCode0 =赤
PixelCode1 =赤
PixelCode2 =赤
PixelCode3 =赤
(b)
第1段目 DS0内のPCS
Pallet_only_update=0
Pallet_id=0
Object_id_ref=1
第2段目 DS1内のPCS
Pallet_only_update=1
Pallet_id=1
Object_id_ref=1
第3段目 DS2内のPCS
Pallet_only_update=1
Pallet_id=2
Object_id_ref=1
第4段目 DS3内のPCS
Pallet_only_update=1
Pallet_id=3
Object_id_ref=1

図27

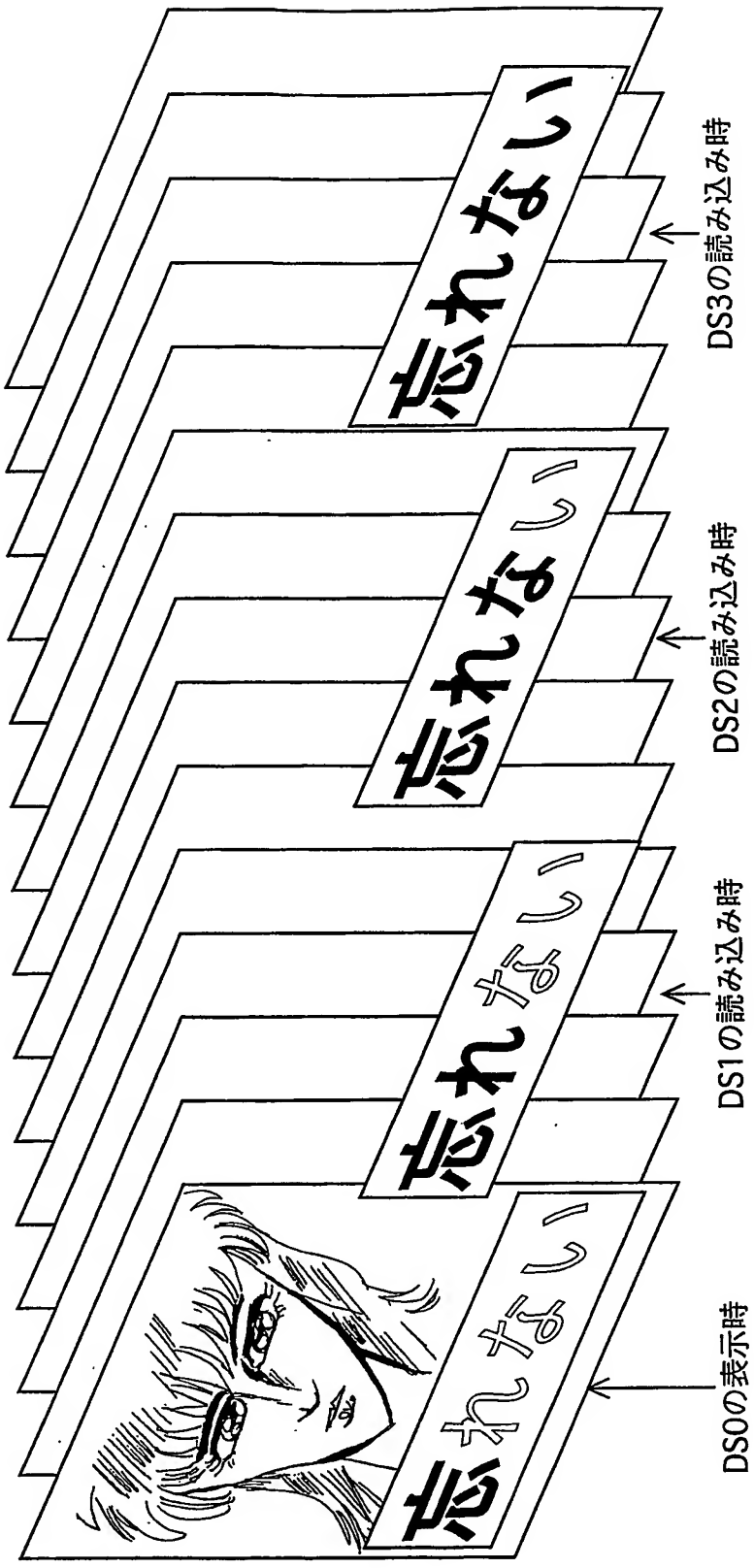

負けない
負けない
負けない
負けない
DS3の読み込み時
DS2の読み込み時
DS1の読み込み時
DS0の表示時

図28

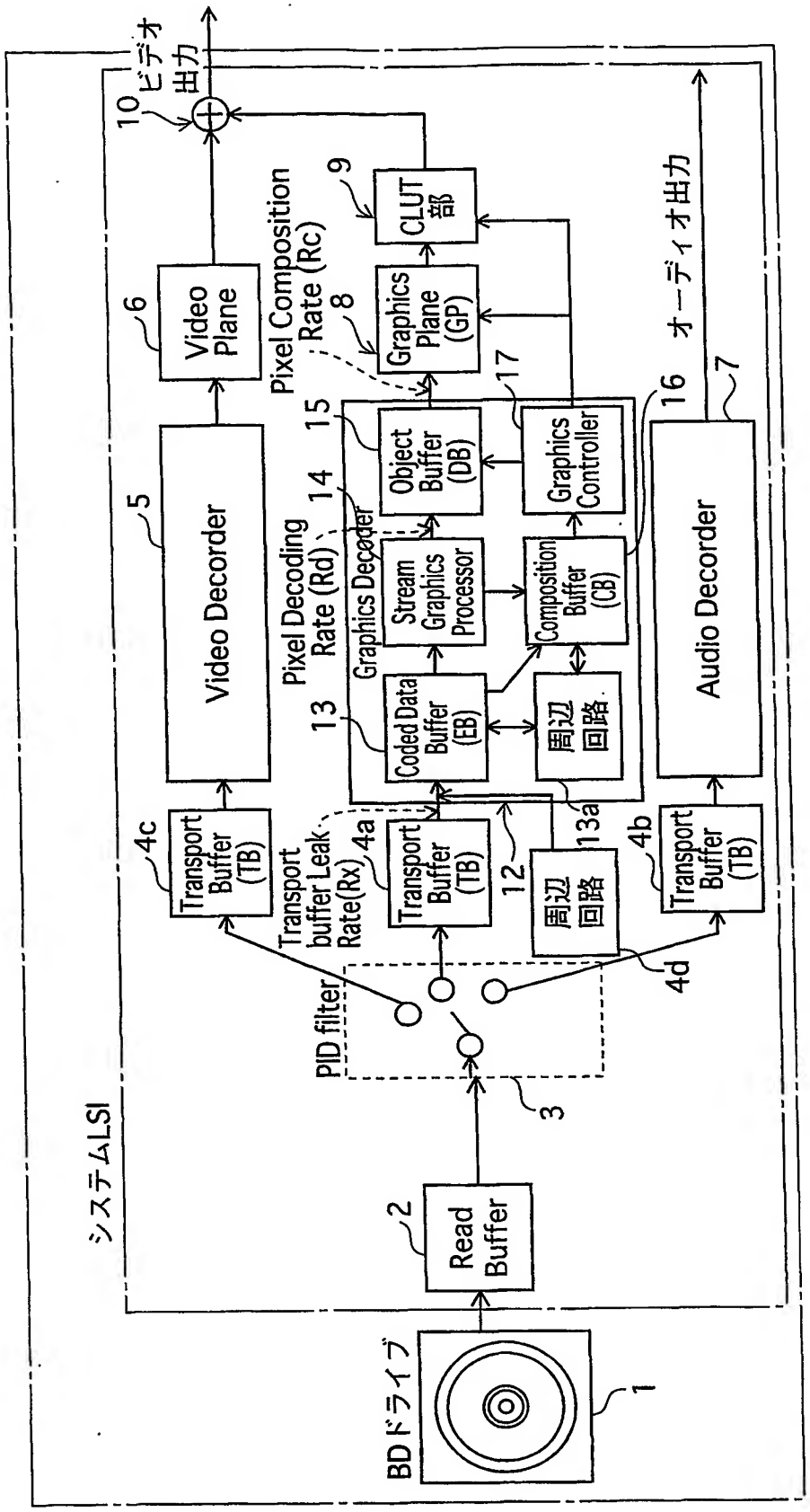

BDドライブ
1
システムLSI
Read Buffer
2
PID filter
3
Transport Buffer (TB)
4a
Transport Buffer (TB)
4b
Transport Buffer (TB)
4c
周辺回路
4d
Video Decorder
5
Video Plane
6
Audio Decorder
7
Graphics Plane (GP)
8
CLUT部
9
ビデオ出力
10
オーディオ出力
16
Graphics Controller
17
Object Buffer (DB)
15
Stream Graphics Processor
14
Graphics Decoder
Coded Data Buffer (EB)
13
周辺回路
13a
Composition Buffer (CB)
12
Pixel Decoding Rate (Rd)
Pixel Composition Rate (Rc)
Transport buffer Leak Rate(Rx)

図29

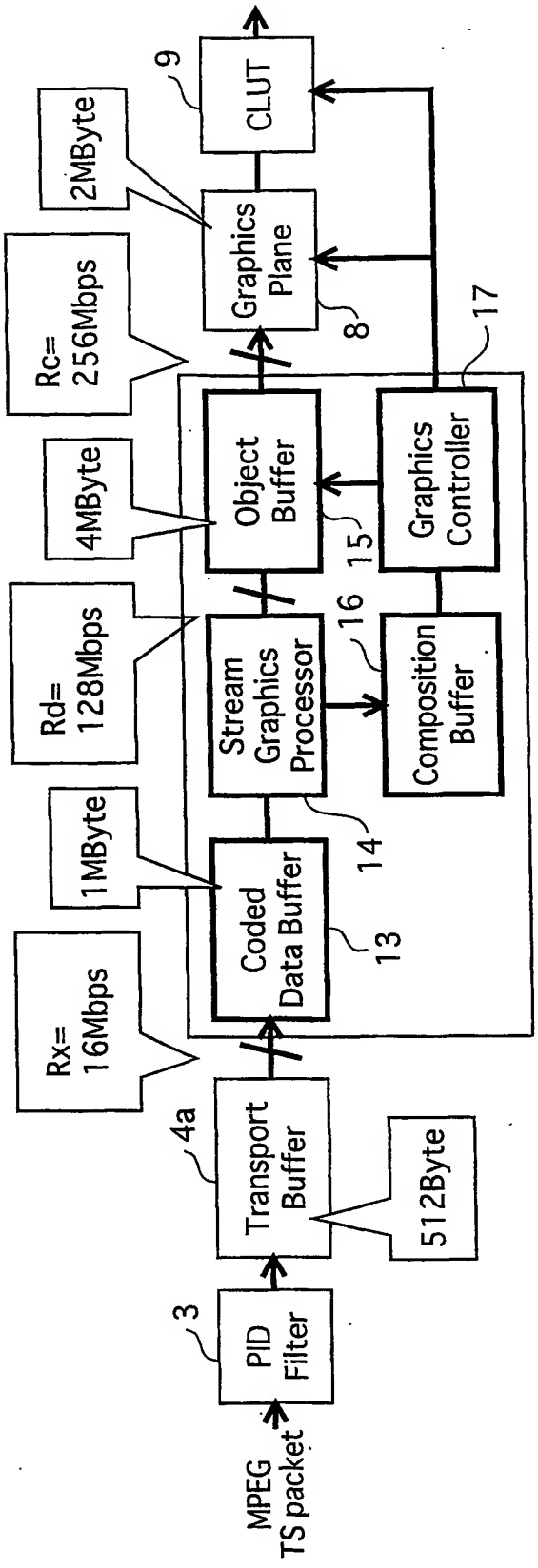

MPEG TS packet
PID Filter
3
Transport Buffer
4a
512Byte
Rx= 16Mbps
Coded Data Buffer
13
1MByte
14
Stream Graphics Processor
16
Composition Buffer
Rd= 128Mbps
Object Buffer
15
Graphics Controller
17
4MByte
Rc= 256Mbps
Graphics Plane
8
2MByte
CLUT
9

図30

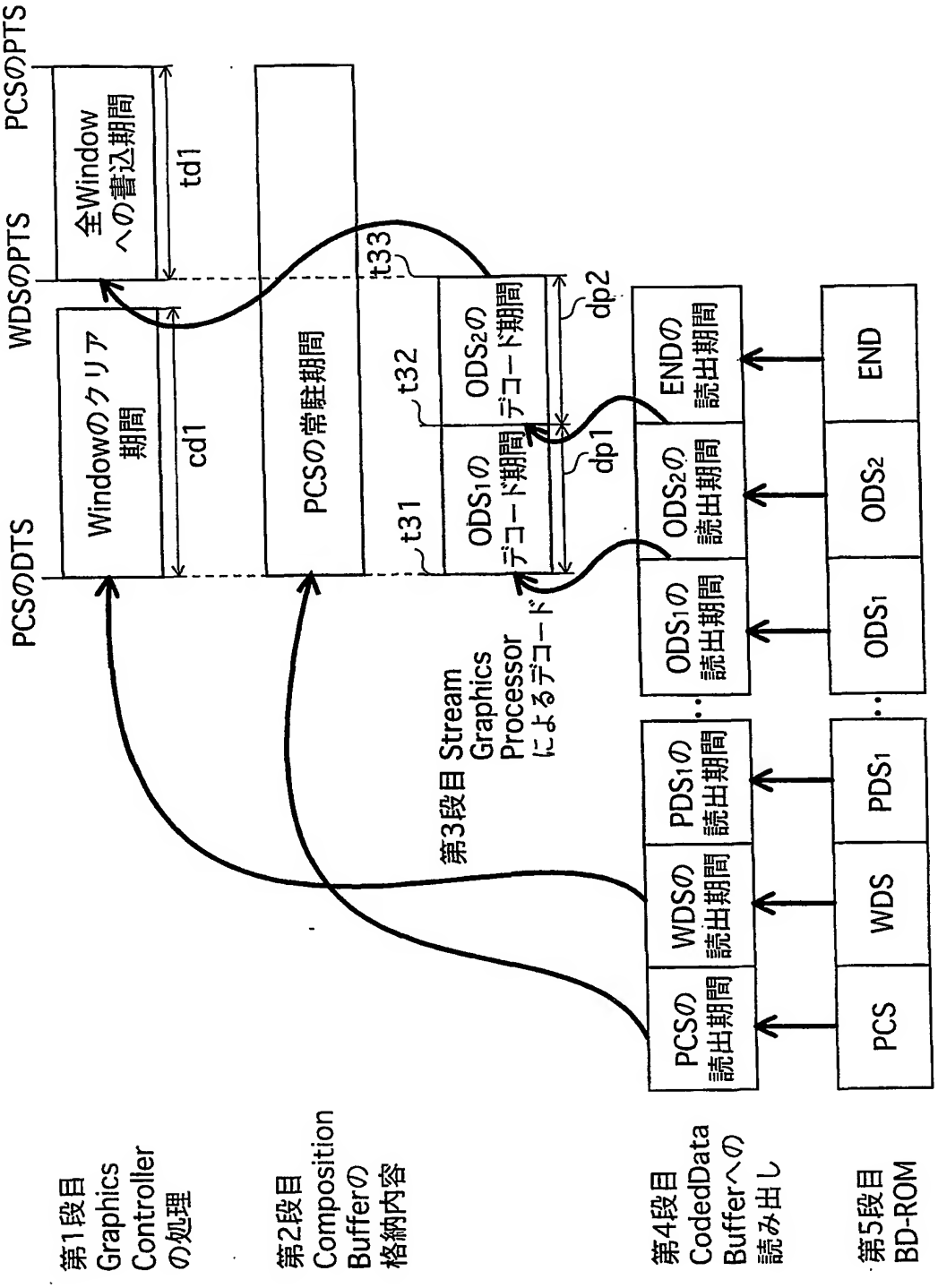

PCSのPTS
WDSのPTS
PCSのDTS
全Windowへの書込期間
td1
Windowのクリア期間
cd1
PCSの常駐期間
t33
t32
t31
ODS2のデコード期間
ODS1のデコード期間
dp2
dp1
第3段目 Stream Graphics Processorによるデコード
END
ODS2
ODS1
PCS
ENDの読出期間
ODS2の読出期間
ODS1の読出期間
WDS
PDS1
WDSの読出期間
PDS1の読出期間
PCSの読出期間
第1段目 Graphics Controllerの処理
第2段目 Compositon Bufferの格納内容
第4段目 CodedData Bufferへの読み出し
第5段目 BD-ROM

図31

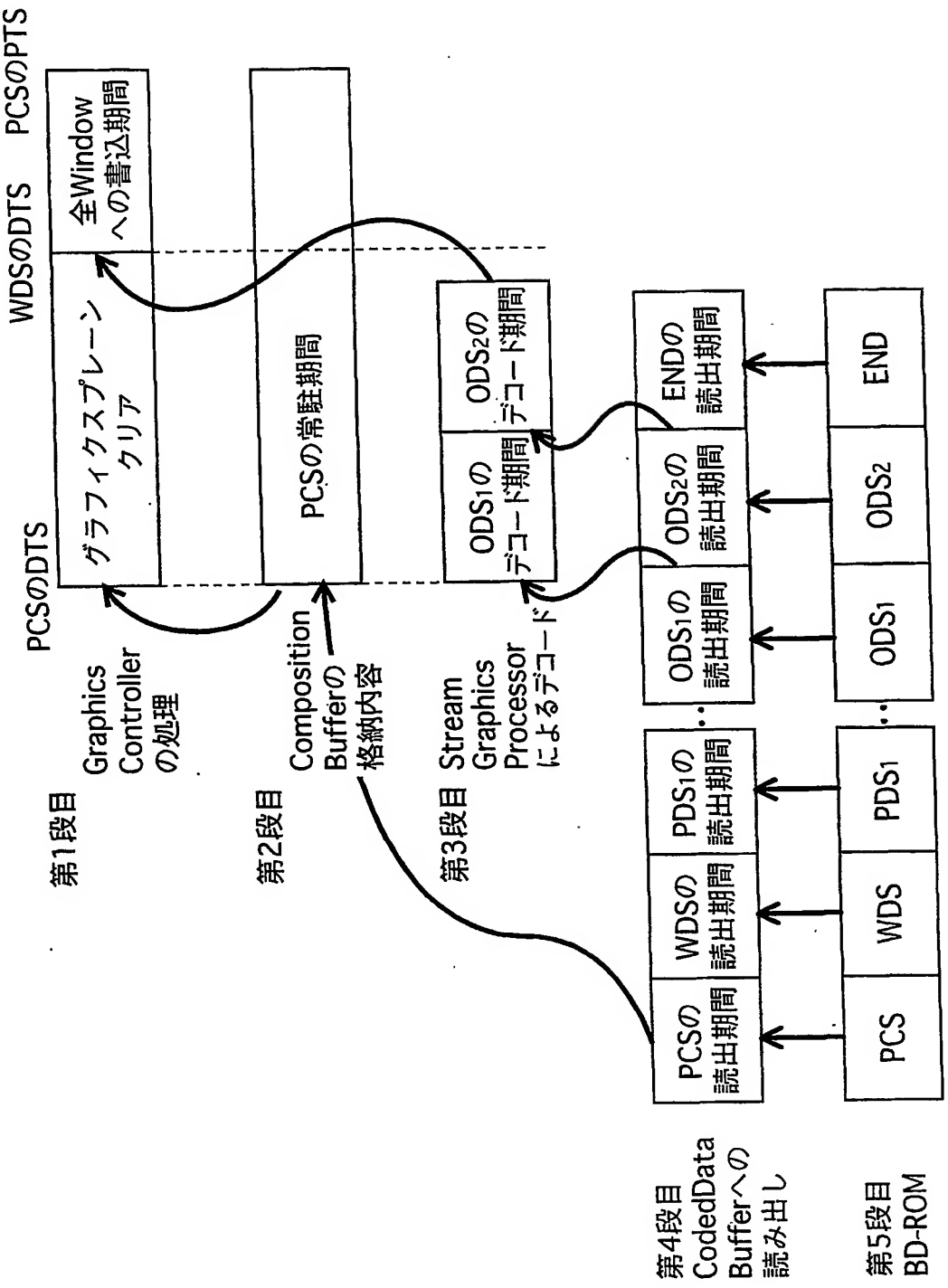

PCSのPTS
WDSのDTS
PCSのDTS
全Windowへの書込期間
グラフィクスプレーンクリア
PCSの常駐期間
ODS2のデコード期間
ODS1のデコード期間
ENDの読出期間
ODS2の読出期間
ODS1の読出期間
END
ODS2
ODS1
PDS1の読出期間
WDSの読出期間
PCSの読出期間
PDS1
WDS
PCS
第1段目
Graphics Controllerの処理
第2段目
Compositionの格納内容
第3段目
Stream Graphics Processorによるデコード
第4段目
CodedData Bufferへの読み出し
第5段目
BD-ROM

図32

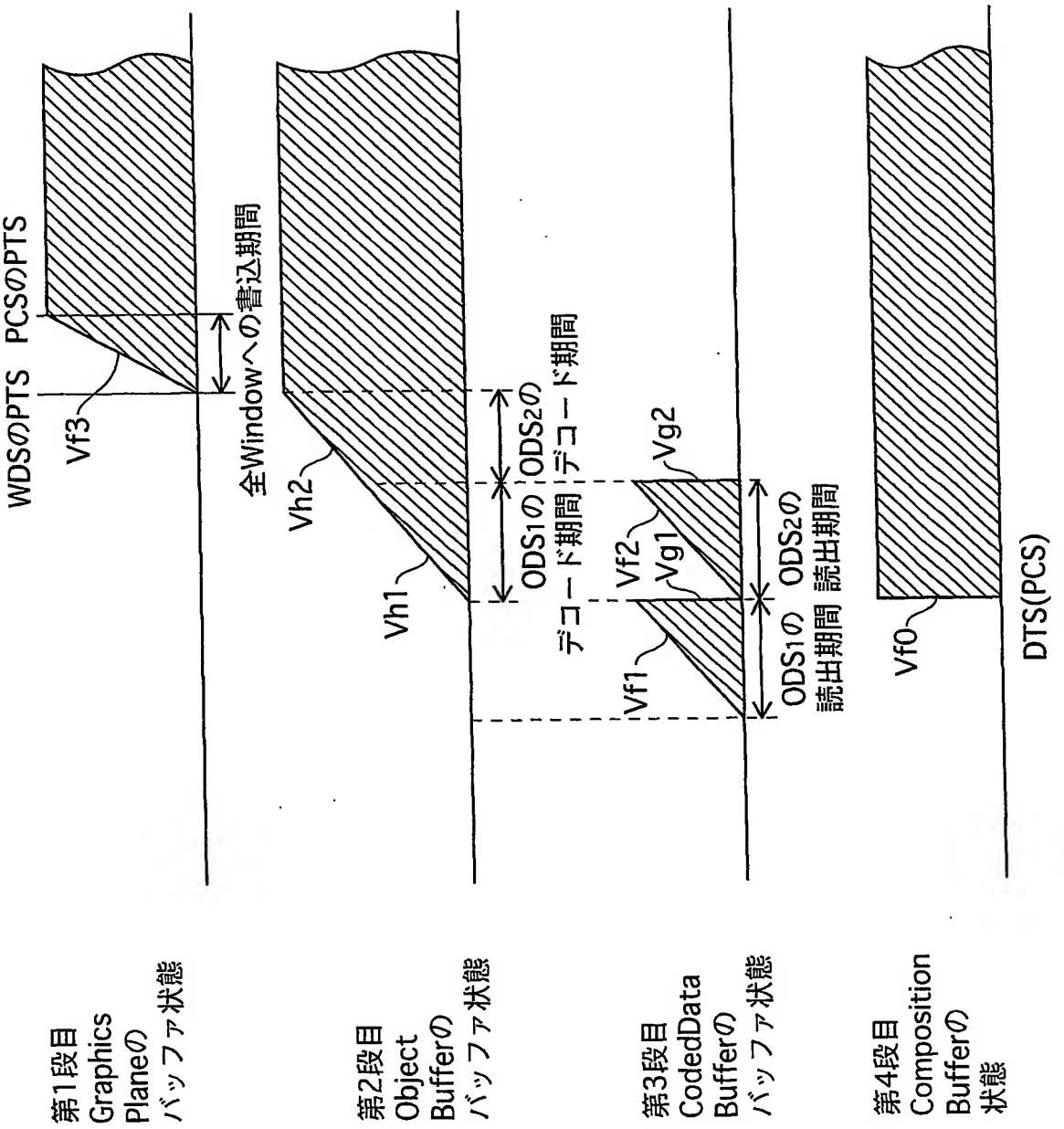

WDSのPTS
PCSのPTS
Vf3
全Windowへの書込期間
Vh2
Vh1
ODS1の
デコード期間
ODS2の
デコード期間
Vf2
Vg1
Vg2
Vf1
ODS1の
読出期間
ODS2の
読出期間
Vf0
DTS(PCS)
第1段目
Graphics
Planeの
バッファ状態
第2段目
Object
Bufferの
バッファ状態
第3段目
CodedData
Bufferの
バッファ状態
第4段目
Composition
Bufferの
状態

図33

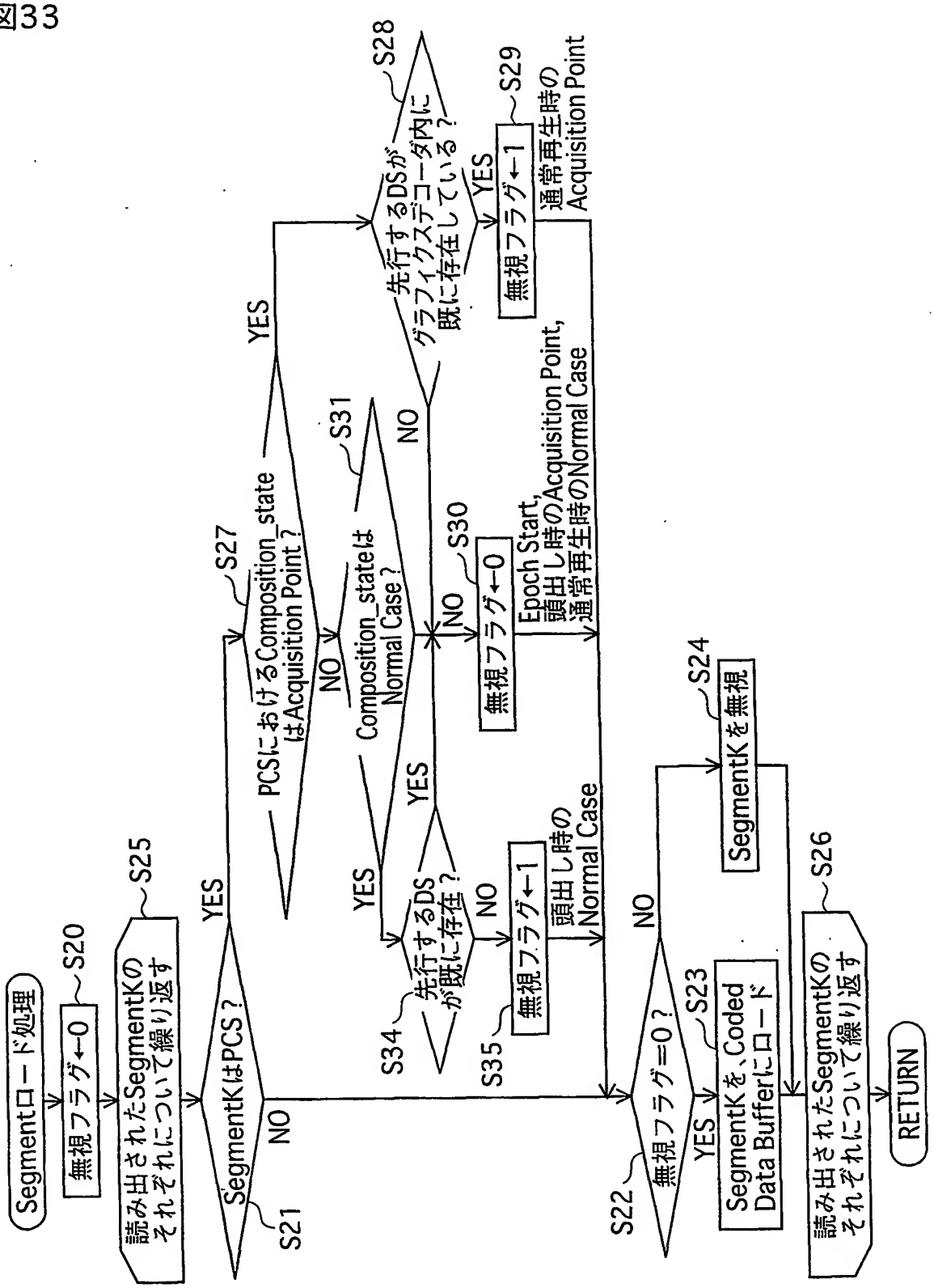

Segmentロード処理
S20
無視フラグ←0
S25
読み出されたSegmentKの
それぞれについて繰り返す
S21
SegmentKはPCS？
YES
NO
S27
PCSにおけるComposition_state
はAcquisition Point？
YES
NO
S31
Composition_stateは
Normal Case？
YES
NO
S28
先行するDSが
グラフィクスデコーダ内に
既に存在している？
YES
NO
S29
無視フラグ←1
通常再生時の
Acquisition Point
S30
無視フラグ←0
Epoch Start,
頭出し時のAcquisition Point,
通常再生時のNormal Case
S34
先行するDS
が既に存在？
YES
NO
S35
無視フラグ←1
頭出し時の
Normal Case
S22
無視フラグ=0？
YES
NO
S23
SegmentKを、Coded
Data Bufferにロード
S24
SegmentKを無視
S26
読み出されたSegmentKの
それぞれについて繰り返す
RETURN

図34

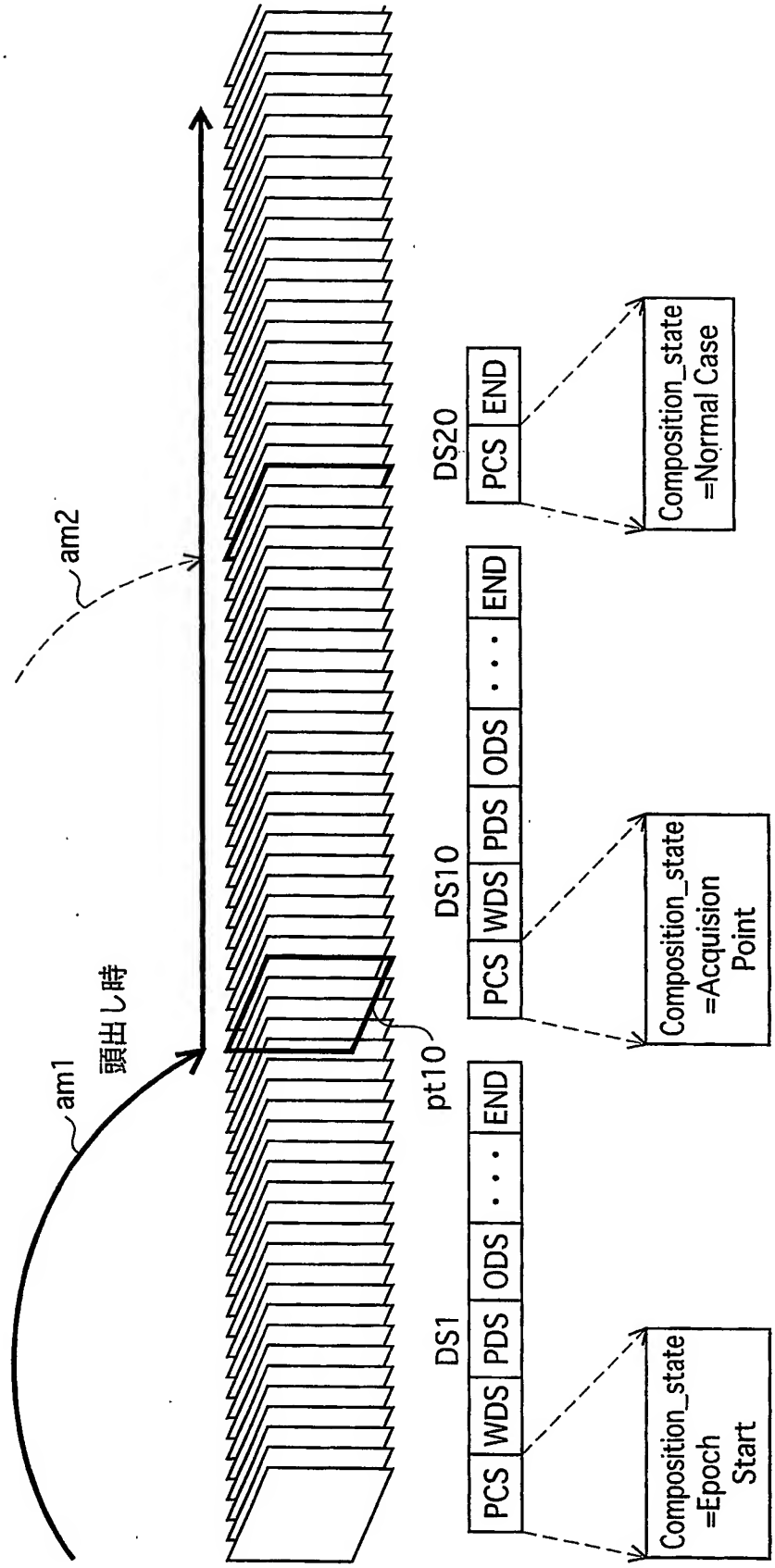

am2
am1
頭出し時
pt10
DS20
PCS
END
Composition_state
=Normal Case
DS10
PCS
WDS
PDS
ODS
・・・
END
Composition_state
=Acquision
Point
DS1
PCS
WDS
PDS
ODS
・・・
END
Composition_state
=Epoch
Start

図35

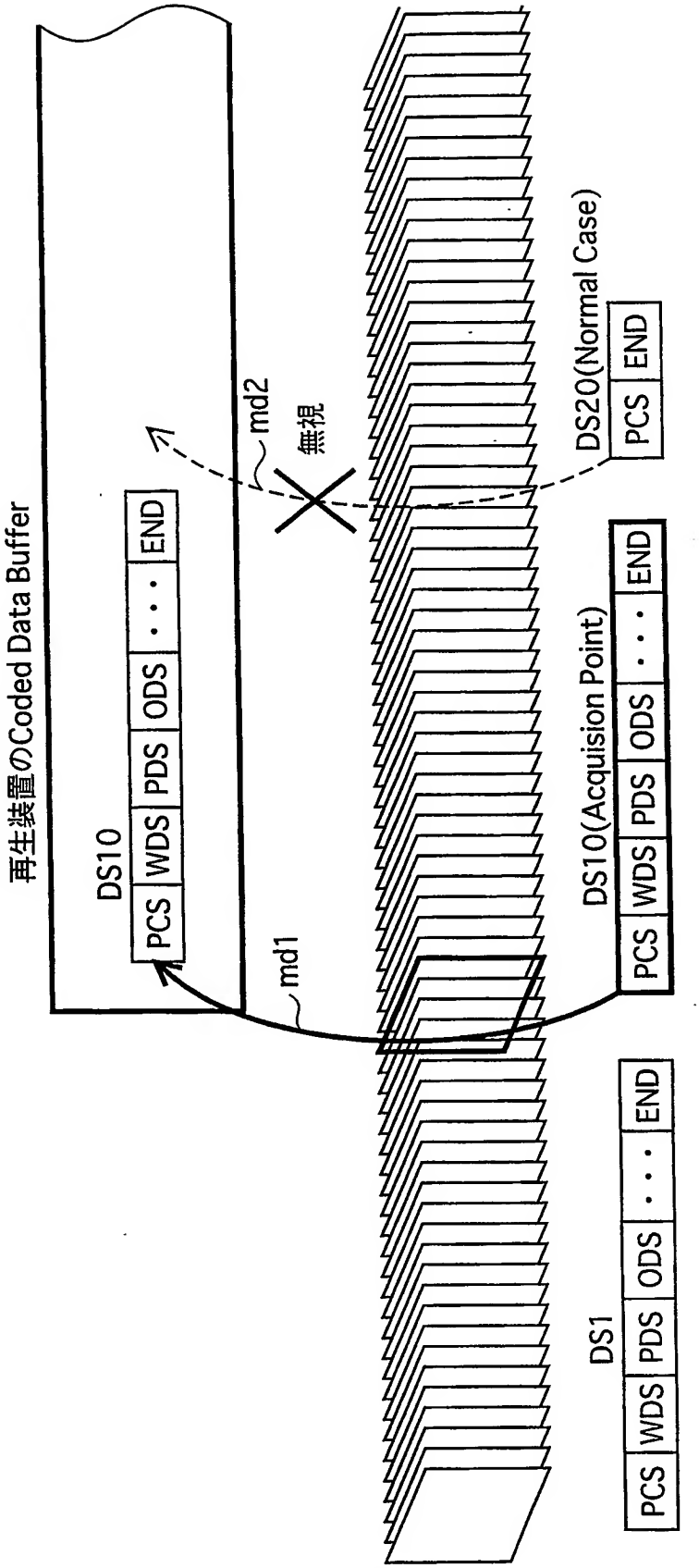

再生装置のCoded Data Buffer
DS10
PCS
WDS
PDS
ODS
・・・
END
md1
md2
無視
DS20(Normal Case)
PCS
END
DS10(Acquision Point)
PCS
WDS
PDS
ODS
・・・
END
DS1
PCS
WDS
PDS
ODS
・・・
END

図36

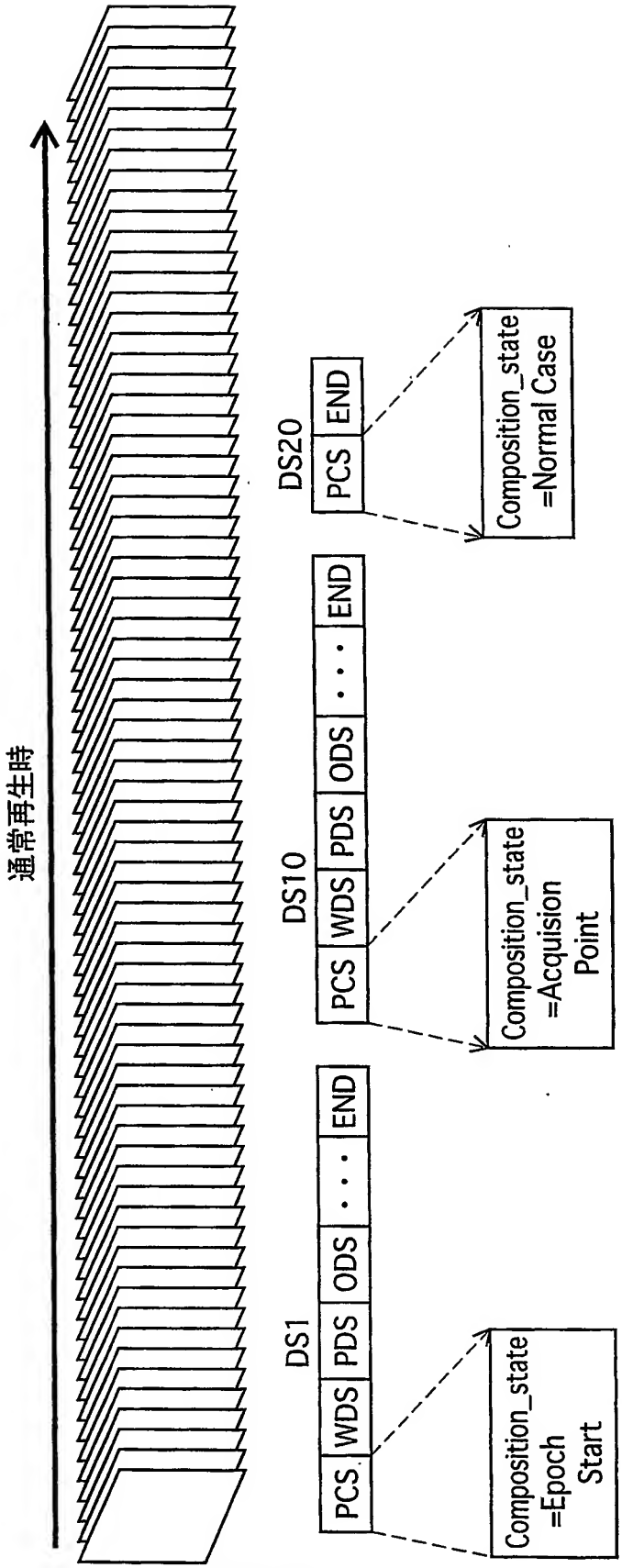

通常再生時
DS1
PCS
WDS
PDS
ODS
・・・
END
Composition_state
=Epoch
Start
DS10
PCS
WDS
PDS
ODS
・・・
END
Composition_state
=Acquision
Point
DS20
PCS
END
Composition_state
=Normal Case

図37

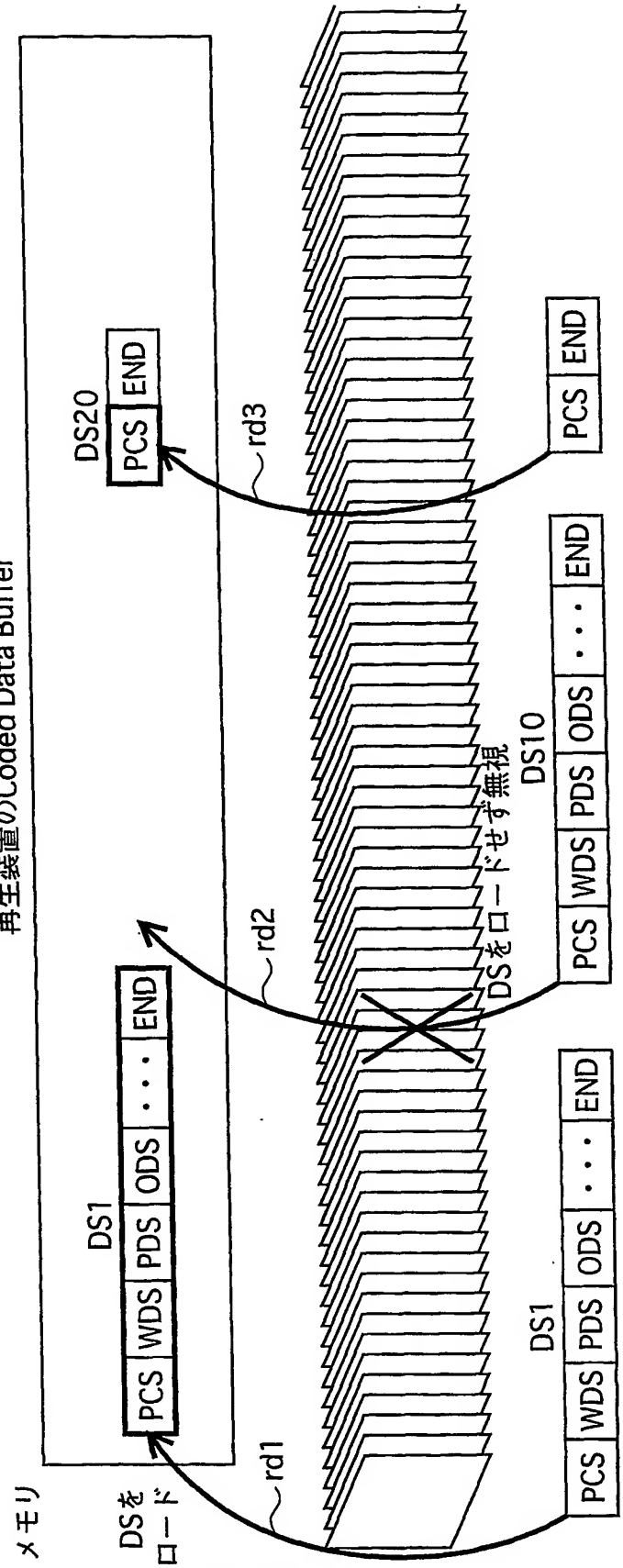

メモリ
DSをロード
再生装置のCoded Data Buffer
DS1
PCS
WDS
PDS
ODS
・・・
END
DS20
PCS
END
rd1
rd2
rd3
DSをロードせず無視
DS10
PCS
WDS
PDS
ODS
・・・
END

図38

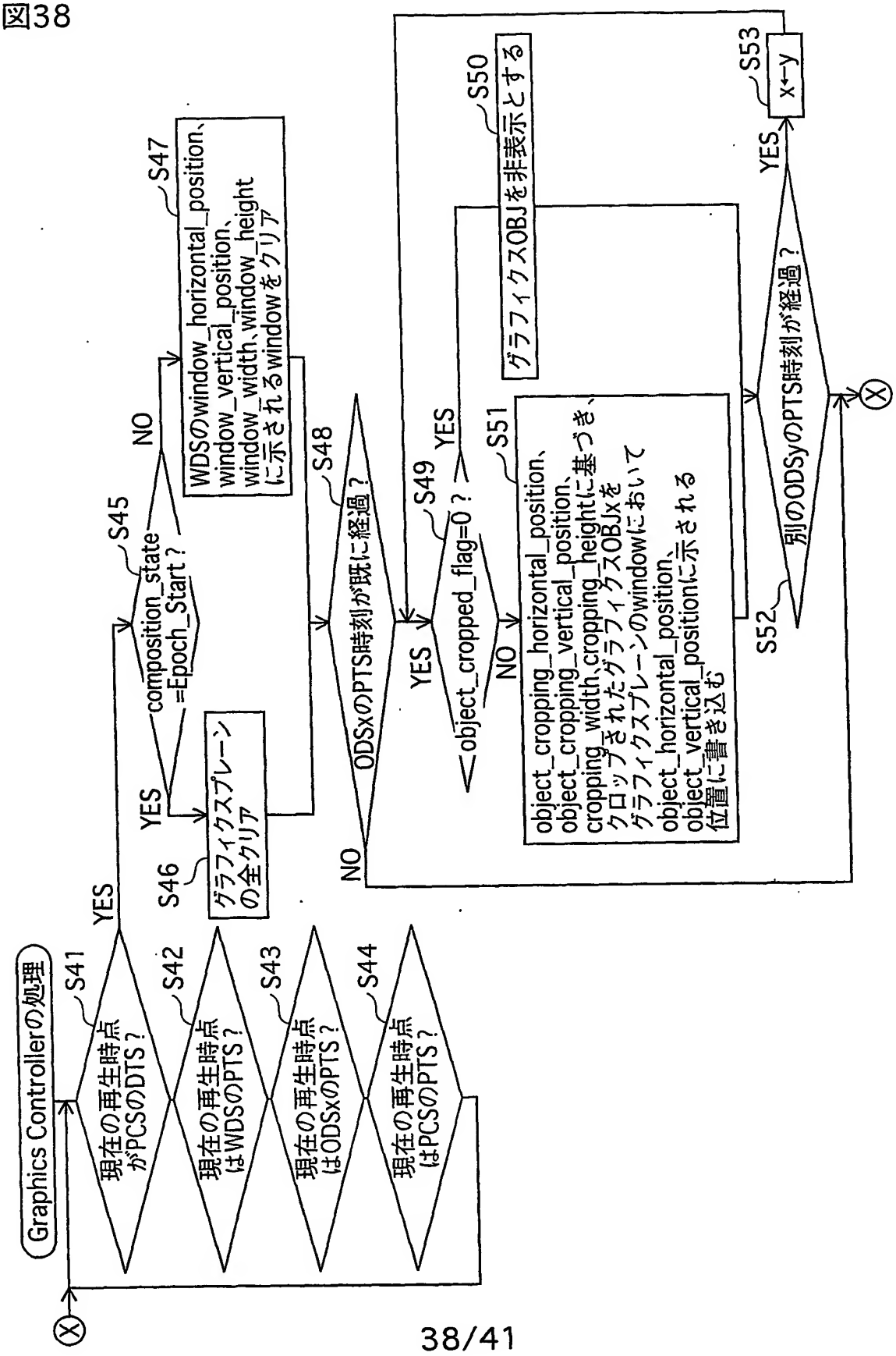

Graphics Controllerの処理
S41
現在の再生時点がPCSのDTS？
YES
S42
現在の再生時点はWDSのPTS？
YES
S43
現在の再生時点はODSxのPTS？
YES
S44
現在の再生時点はPCSのPTS？
YES
S45
composition_state=Epoch_Start？
NO
S47
WDSのwindow_horizontal_position、window_vertical_position、window_width、window_heightに示されるwindowをクリア
YES
S46
グラフィクスプレーンの全クリア
S48
ODSxのPTS時刻が既に経過？
NO
YES
S49
object_cropped_flag=0？
YES
S50
グラフィクスOBJを非表示とする
NO
S51
object_cropping_horizontal_position、object_cropping_vertical_position、cropping_width、cropping_heightに基づき、クロップされたグラフィクスOBJxをグラフィクスプレーンのwindowにおいてobject_horizontal_position、object_vertical_positionに示される位置に書き込む
S52
別のODSyのPTS時刻が経過？
YES
S53
x←y

図39

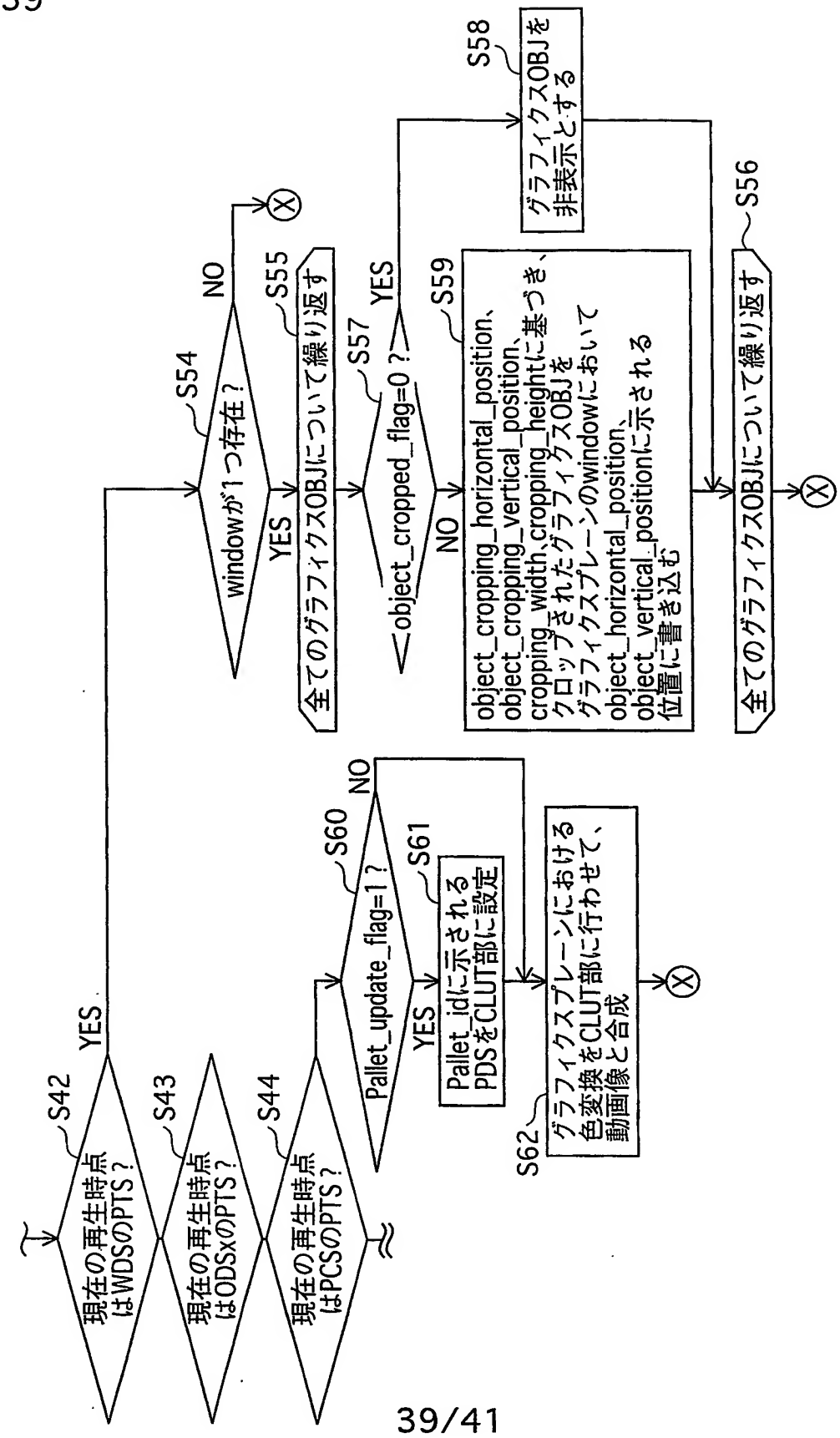

現在の再生時点はWDSのPTS？
現在の再生時点はODSxのPTS？
現在の再生時点はPCSのPTS？
YES
S42
S43
S44
Pallet_update_flag=1？
NO
YES
S60
S61
Pallet_idに示されるPDSをCLUT部に設定
グラフィクスプレーンにおける色変換をCLUT部に行わせて、動画像と合成
S62
windowが1つ存在？
NO
YES
S54
全てのグラフィクスOBJについて繰り返す
S55
object_cropped_flag=0？
YES
NO
S57
S58
グラフィクスOBJを非表示とする
S59
object_cropping_horizontal_position、object_cropping_vertical_position、cropping_width、cropping_heightに基づき、クロップされたグラフィクスOBJをグラフィクスプレーンのwindowにおいてobject_horizontal_position、object_vertical_positionに示される位置に書き込む
全てのグラフィクスOBJについて繰り返す
S56

図40

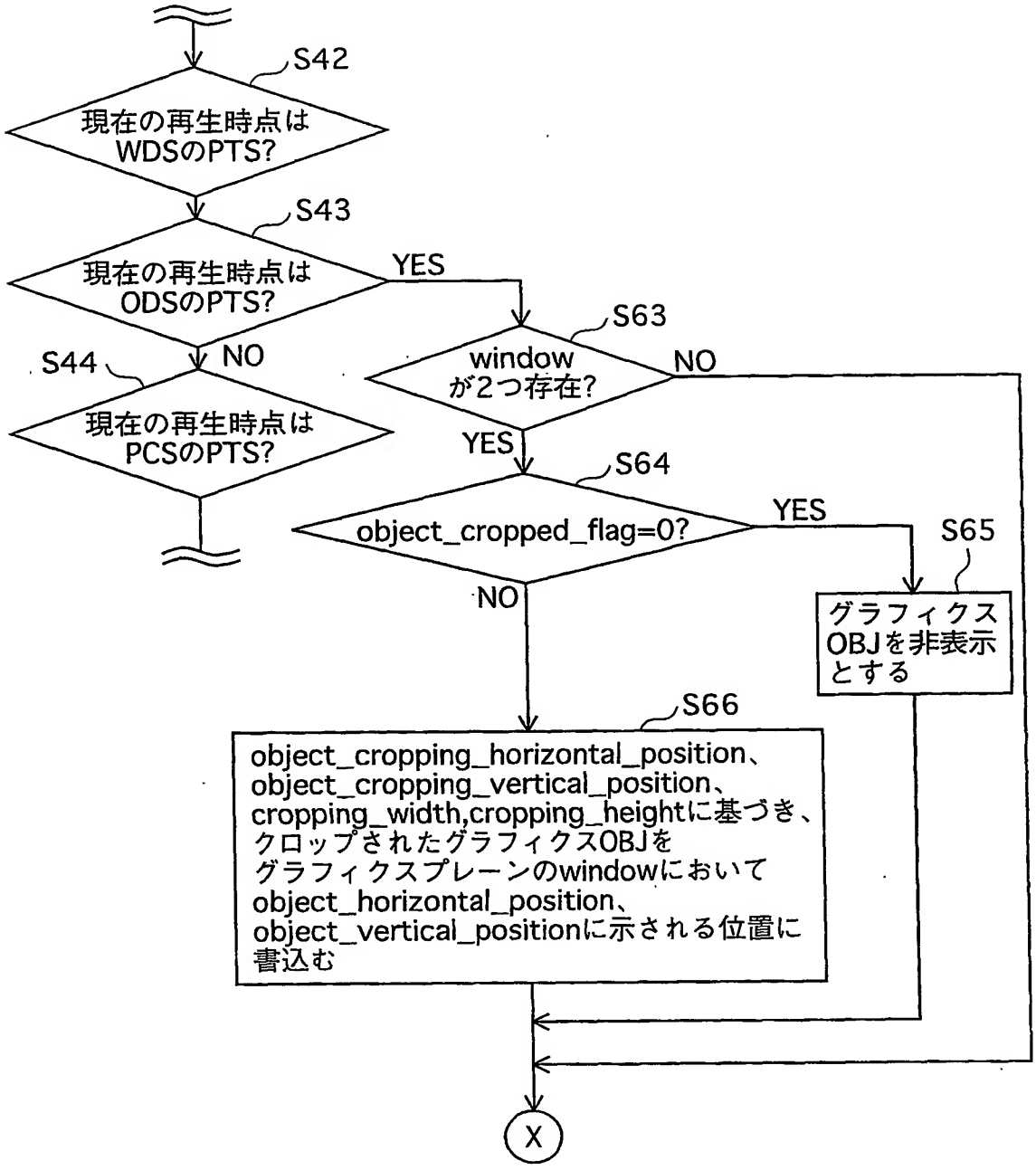
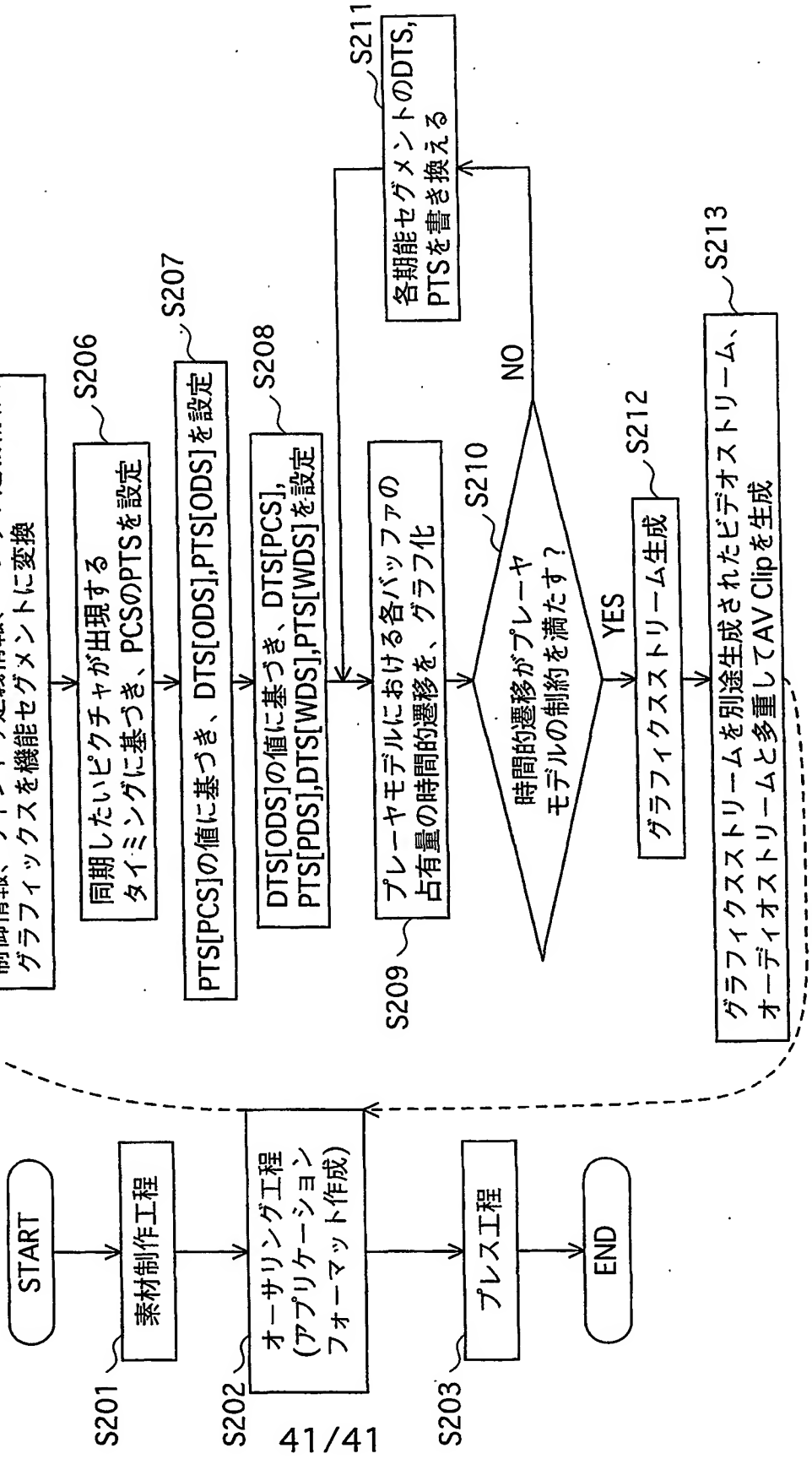

S42
現在の再生時点は
WDSのPTS?
S43
現在の再生時点は
ODSのPTS?
YES
S44
NO
現在の再生時点は
PCSのPTS?
S63
window
が2つ存在?
NO
YES
S64
object_cropped_flag=0?
YES
S65
グラフィクス
OBJを非表示
とする
NO
S66
object_cropping_horizontal_position、
object_cropping_vertical_position、
cropping_width,cropping_heightに基づき、
クロップされたグラフィクスOBJを
グラフィクスプレーンのwindowにおいて
object_horizontal_position、
object_vertical_positionに示される位置に
書込む
X

図41


START
素材制作工程
S201
オーサリング工程
(アプリケーション
フォーマット作成)
S202
プレス工程
S203
END
制御情報、ウィンドウ定義情報、
パレット定義情報、グラフィックスを記述
S204
制御情報、ウィンドウ定義情報、パレット定義情報、
グラフィックスを機能セグメントに変換
S205
同期したいピクチャが出現する
タイミングに基づき、PCSのPTSを設定
S206
PTS[PCS]の値に基づき、DTS[ODS],PTS[ODS]を設定
S207
DTS[ODS]の値に基づき、DTS[PCS],
PTS[PDS],DTS[WDS],PTS[WDS]を設定
S208
プレーヤモデルにおける各バッファの
占有量の時間的遷移を、グラフ化
S209
時間的遷移がプレーヤ
モデルの制約を満たす？
S210
NO
YES
各期能セグメントのDTS,
PTSを書き換える
S211
グラフィクスストリーム生成
S212
グラフィクスストリームを別途生成されたビデオストリーム、
オーディオストリームと多重してAV Clipを生成
S213